VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Analýza vztahů webových stránek a použitých vzorů

Analysis of relations between web pages and used patterns

2011 Bc. Mojmír Šmída

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 4. května 2011 .....

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

Vedoucí práce: Mgr. Miloš Kudělka, Ph.D.

Abstrakt

Cílem práce je implementace metod pro automatickou detekci vybraných webových vzorů na webových stránkách a provedení experimentů vycházejících z popisu webových stránek pomocí návrhových vzorů. V teoretické části uvedu přehled metod využitelných pro získání informací z webových stránek. Metody, které implementuji, vycházejí z Gestalt principů (blízkost, podobnost, souvislost, celek). Práce se zabývá analýzou domény, výběrem často se vyskytujících návrhových vzorů a sestavením klíčových slov. Klíčová slova jsem vybíral ručně. V závěru práce jsem provedl testování a vyhodnocení výsledků.

Klíčová slova: automatická detekce, webové vzory, webové stránky, klíčová slova, dolování textu, dolování webu, kategorizace

Abstract

The goal is the implementation of methods for automatic detection of selected web patterns on the web pages and design experiments based on the description of Web pages using design patterns. In the theoretical part I will review the methods available for obtaining information from the website. Methods that implement are based on Gestalt principles (proximity, similarity, continuity, closure). The paper analyzes the domain by selecting frequently occurring design patterns and the establishment of key words. Key words I chose manually. Finally, I carried out testing and evaluation of results.

Keywords: automatic detection, web patterns, web site, keywords, text mining, web mining, categorization

Seznam použitých zkratek a symbolů

DIS	–	Dokumentografické informační systémy
DM	–	Data Mining
HAP	–	HTML Agility Pack
HTML	–	Hyper Text Markup Language
IDF	–	Inverse Document Frequency
ILP	–	Induktivní Logické Programování
IR	–	Information Retrieval
KS	–	Klíčová Slova
SEO	–	Search Engine Optimization
SVD	–	Singular Value Decomposition
TF	–	Term Frequency
TFIDF	–	Term Frequency Inverse Document Frequency
TM	–	Text Mining
W3C	–	World Wide Web Consortium
WWW	–	World Wide Web
XML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Dnešní svět je charakterizován explozí objemu dat. Disky pojmou stále více dat, a proto neustále vzrůstá i objem ukládaných dat, ať už užitečných nebo zbytečných. Je pravděpodobné, že v těchto datech je ukryto mnohem více informací, než které můžeme jednoduše vyčíst. Zpracování dat z rozsáhlých databází má v dnešní Informatice nejrůznější formy. Tradiční přístupy analyzují data prostřednictvím dotazovacích nástrojů SQL. Přesto existuje mnoho úloh, na které tyto běžné přístupy nestačí.

V současné době je Internet nejpoužívanějším zdrojem pro získání informací. Aby jsme získali relevantní informace, potřebujeme k tomu tzv. vyhledávač. Například Google[1] nebo Seznam[2]. Více v indexu search engines[3]. Pro správnou funkci potřebuje vyhledávač měřit kvalitu stránek pomoci tzv. Ranku (např. PageRank u Google, S-Rank u Seznam). Obecně vyhledávač pracuje ve třech krocích:

1. procházení webových stránek
2. vytvoření databáze výskytu slov
3. indexování

Procházení zajišťuje program tzv. vyhledávací robot, který prochází stránky přes hypertextové odkazy a snaží se navštívit ideálně celý Internet. Stránky si ukládá do databáze a po čase se k nim vrací, aby sledoval změny. Z uložených stránek robot vybere všechna slova, která uloží do databáze. Ke každému slovu si pamatuje stránku kde se vyskytuje. Tím vzniká velice obsáhlý rejstřík. K urychlení vyhledání se používá indexování. Index se tvoří tak, aby se zobrazily na prvních místech stránky s největší relevancí. Pro výpočet relevance se používají různé metody analýzy. Např. váha slov (slovo se vyskytuje v titulku, frekvence slova, hustota), atraktivita (na stránku směřuje více odkazů), sponzorované odkazy (zaplacením poplatku se zvyšuje váha) nebo technická kvalita (váha se zvyšuje pokud stránky vyhovují W3C[4] standardům). Jiný způsob jak dostat web na přední pozice je optimalizace pomocí SEO. Více o SEO zde [1].

Jiný přístup poskytuje meta vyhledávač Pattrio[5]. Jiný v tom smyslu, že používá metodu založenou na sémantice webových stránek. Pracuje jak se strukturou, tak i s textovým obsahem stránky. V následujících větách velice stručně objasním jak funguje Pattrio. Vyhledávač získá stránku (například přes Google), která obsahuje slova zadaná

[1] http://www.google.cz
[2] http://www.seznam.cz
[3] http://www.searchenginesindex.com
[4] http://www.w3.org
[5] http://www.pattrio.net

uživatelem. Z tohoto dokumentu vybere pouze text. V textu hledá určité úseky, která splňují kritéria. Tyto kritéria jsou popsané ve slovníku návrhových vzorů (knihovna návrhových vzorů[6]). V předchozím kroku jsme textu přiřadili sémantiku. Jestliže pattrio rozhodl, že se jedná o návrhový vzor, využije této skutečnosti a zobrazí ji ve výpisu relevantních stránek. Tímto získáme další užitečné informace o stránce.

Zpracováním velkého množství dat se zabývá oblast nazvaná *Information Retrieval*. Tímto zpracováním je myšleno ukládání, analýzu, vyhledávání, kategorizace, segmentace a sumarizace. Data, ve kterých vyhledávám jsou dokumenty. Tyto dokumenty prezentují základní jednotku textu. Rozsah dokumentů muže být různý. V této práci se zabývám pouze webovými stránkami, ale obecně se jedná o literární díla, novinové články či dokonce zvuk nebo video. Výsledkem vyhledávání je množina dokumentů, která odpovídá uživatelskému dotazu.

Cílem této práce je vytvořit aplikaci, která bude schopna analyzovat a rozhodovat, zda webová stránka obsahuje relevantních informace. Cílem je sestavit vlastní množinu návrhových vzorů a na těch provádět experimenty. Pro experimenty jsem shromáždil kolekci webových stránek. Při tvorbě této práce bylo potřeba získat znalosti z několika oblastí a řešit různé problémy. To jsem shrnul do několika bodů.

- Návrhový vzor
- DIS (Modely používané pro vyhledávání dokumentů)
- Data mining, Text mining, Web mining
- Návrh vlastní metody řešení
- Návrh a implementace aplikace
- Experimenty

Struktura práce se bude držet těchto bodů. V kapitole 2 vysvětlím co je návrhový vzor[7].

V kapitole DIS uvádím dva modely. První z nich je model booleovský. Tento model mohu využít pro popis a vyhledání dokumentu, ve kterém jsem již objevil (neobjevil) návrhový vzor. U takového dokumentu udržuji seznam, který obsahuje název návrhového vzoru a hodnotu ano/ne. Pomocí tohoto seznamu mohu vyhledat jen ty dokumenty, které obsahují určité návrhové vzory. Druhý z modelů je model vektorový, který popíšu v kapitole 3.2. Vektorový model využívám k popisu dokumentů a vyhodnocení podobnosti s dotazem. Dokument popisuji pomocí návrhových vzorů. Ve

[6] http://www.welie.com/patterns/

[7] návrhový vzor není totéž jako vzor u Data Miningu

svojí práci ohodnocuji váhu slova (návrhového vzoru) podle metody TF (počet výskytů v dokumentu). Jiným způsobem hodnocení je metoda TFIDF, která zohledňuje frekvenci termu v ostatních dokumentech. Pomocí této metody bych mohl automaticky vypočítat váhu návrhového vzoru s ohledem na to v kolika jiných dokumentech (jiná doména) se návrhový vzor vyskytuje.

V kapitole 4.1 objasním co je Data mining. Data mining jsem zde uvedl, protože moje práce využívá metody porozumění a predzpracování dat. Kapitola 4.2 pojednávám o Text miningu. Text mining se mimo jiné zabývá zpracováním a přípravou textových dat a přesně tyto problémy řeším v této práci. V poslední teoretické kapitole Web mining jsem popsal použití návrhových vzorů pro popis a klasifikaci webových stránek. To je základní myšlenka, kterou využívá tato práce.

2 Návrhový vzor

V této práci budu používat pojem *návrhový vzor* a *vzor*. Tyto dva termíny nejsou ekvivalentní. Vysvětlím v následujícím textu.

Návrhový vzor (design pattern) chápu jako popis charakteristických strukturálních rysů a rysů chování, které zlepšují použitelnost architektury software, uživatelského rozhraní, webových stránek nebo cokoliv jiného v zamýšlené doméně. V této práci používám návrhové vzory, které obsahují sémantické informace - vzory domén. V následujícím seznamu uvedu základní rysy návrhových vzorů:

- málo závislý na implementaci
- závislý na vnímání uživatele
- komunikační nástroj mezi uživatelem a programátorem
- jednotlivé prvky konkrétní instance vzoru jsou na stránce víceméně pohromadě

Pro lepší představu uvedu konkrétní příklad návrhového voru *Nabídka zájezdu* viz. obrázek 1. Stručný popis: *Název vzoru*: Nabídka dovolené. *Problém*: potřeba přehledného

Obrázek 1: Návrhový vzor Nabídka dovolené

zobrazení informací o zájezdu. *Kdy užít?*: prodej a nabídka zájezdů, tam kde je potřeba zobrazit přehledný seznam nabízených zájezdů. *Proč?*: prodejce potřebuje přehledně a srozumitelně zobrazit informaci o zájezdech. *Jak?*: zobrazení jedné a více položek (typicky 4 a více). *Obvykle obsahuje*: název letoviska, obrázek, termín, počet dní, druh stravy, cenu, tlačítko pro více informací nebo pro objednání.

Vzor používám ve vztahu s Data miningem. Úkolem data miningu je extrakce zajímavých ne triviálních (potencionálně užitečných, skrytých) souvislostí v datech. Výsledkem je vytvoření pravidel chování, která se dlouhodobě opakují. Tato pravidla se nazývají vzor. Příklad vzoru: v minulosti se data chovala podle nějakého vzoru (zákazník, který má věk větší než 30 let a je ženatý, si obvykle kupuje velké rodinné auto), v budoucnu se chování zákazníků nezmění (zákazníkům, kteří mají věk větší než 30 let a jsou ženatí nabízej rodinná auta. Například zasláním akční nabídky na email).

3 Dokumentografické informační systémy

Dokumentografické informační systémy (DIS) představují třídu programových nástrojů, určených pro zpracování a výběr dokumentů. Obor, kterým se DIS zabývá se nazývá Information Retrieval (vyhledávání informaci). Vyhledání informací je činnost, jejímž cílem je výběr relevantních dokumentů. Souvisí s reprezentací, ukládáním a přístupem k informacím.

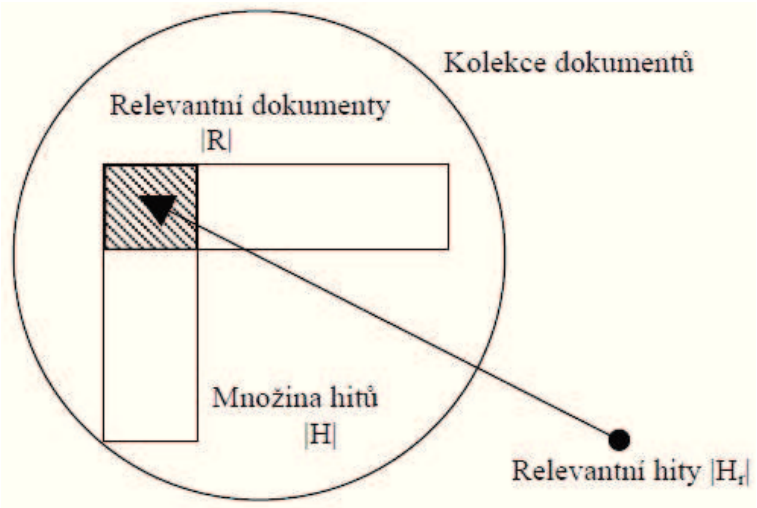
Objekty v Information Retrieval(dále jen IR) jsou vstupní a výstupní. Základním vstupním objektem je dokument. Dokument může být např.: novinový článek, web, fotografie, řeč a zvuk, video. Každý dokument může být popisován pomocí metadat, struktury textu a obsahu. Každý dokument může být popsán pomocí DDL(document description language). Jiný vstupní objekt je dotaz. Dotaz může být cokoliv z výše uvedených. Výstupním objektem je obvykle množina dokumentů. Tyto dokumenty tvoří odpověď na dotaz a nazývají se hity [20],[19].

Tímto se dostáváme k hodnocení efektivity vyhledávacích systémů. Pro změření kvality může být důležitá celá řada faktorů (rychlost, poskytnutí informace o relevantních dokumentech). Schopnost poskytnout informaci o relevantních dokumentech se vyjadřuje pomocí dvou koeficientů. Pojmem relevantní dokument označíme dokument, který vyhovuje svým obsahem dotazu tazatele. Tedy ne všechny dokumenty vybrané jako odpovídající uživatelovu dotazu musí být relevantní a naopak ne všechny relevantní dokumenty musí odpovídat dotazu.[14],[19]. Viz. obrázek 2 na straně 11, který je převzatý z [20].

Popis obrázku 2. $|H_r|$ - počet vybraných relevantních dokumentů, $|R|$ - počet všech relevantních dokumentů v kolekci, $|H|$ - počet všech vybraných dokumentů.

- Koeficient přesnosti = $\frac{|H_r|}{|H|}$
- Koeficient úplnosti = $\frac{|H_r|}{|R|}$

Koeficient přesnosti chápeme jako pravděpodobnost, že vybraný dokument je relevantní. **Koeficient úplnosti** chápeme jako pravděpodobnost s jakou byly vybrány všechny relevantní dokumenty. V ideálním případě by koeficienty měly být rovny 1. Toho v praxi nelze dosáhnout. V praxi tedy uživatel, který po prvním dotazu dostane příliš mnoho nerelevantních dokumentů, začne dotaz zpřesňovat, tím docílí toho, že obdrží více relevantních dokumentů, ale bude jich méně.[14],[20]. V následující kapitole podrobně popíšu základní dva modely. Jsou to Booleovský a Vektorový.



Obrázek 2: Hodnocení efektivity

3.1 Booleovský model

Booleovské DIS pocházejí z 50. let. Jednalo se o první systémy pro automatizaci knihovnictví. Přesto se však tyto systémy ve velké míře používají dodnes (převážně pro jejich snadnou implementaci). Každý dokument D je v indexu reprezentován pomocí množiny termů $T = (t_1, \ldots, t_m)$, které jej co nejlépe popisují. Dokument $D \subseteq T$. Přiřazení množiny termů každému dokumentu probíhá dvěma způsoby:

- **Ruční** - Nekonzistentní. Více lidí, kteří ohodnocují dokument se nemusí shodnout na množině termů. Jeden člověk může ohodnotit stejný dokument jinak - subjektivní pohled.
- **Automatická** - Konzistentní, ale chybí lidskost - porozumění textu (synonyma).

Množina termů je buď předem daná a vyberou se ty, které jsou vhodné pro dokument, nebo se množina termů mění s přibývajícími dokumenty. V booleovském modelu je zajímavě vytvářen index. Ten totiž není tvořen tak, že ke každému dokumentu je přiřazena množina termů, které obsahuje, ale naopak, ke každému termu je přiřazena množina dokumentů, které daný term obsahují. Tohoto se využívá při hledání dokumentů (např.: průnik množin dokumentů, které obsahují slovo A a množina dokumentů obsahující slovo B)

Dotaz je pak sestaven z termů a logických spojek dávající logický výraz. Obecně lze použít následující logické spojky:

- $A \bigwedge B$: Logický součin, konjunkce. Ve výsledku budou dokumenty, které obsahují A a zároveň B.
- $A \bigvee B$: Logický součet, disjunkce. Ve výsledku budou dokumenty, které obsahují A nebo B.
- $A \bigoplus B$: Exkluzivní součet. XOR. Ve výsledku budou dokumenty, které obsahují A nebo B, ne však oba současně.
- $\neg A$: Negace. Ve výsledku budou dokumenty, které neobsahují A.

Protože použitím výše uvedených pravidel by informační systém moc kvalitní nebyl, jsou používány další rozšíření:

- Sekundární informace: *databáze* $\bigwedge$ (autor=*Šarmanová*).
- Zástupné znaky: * - pro více znaků, *?* - pro jeden znak
- Proximitní omezení:
  - A (m,n) B: slovo A je vzdáleno minimálně m a maximálně n slov od termu B.
  - A věta B: term A se vyskytuje ve stejné větě jako term B (stejně pro odstavec a kapitolu).

Problém booleovského modelu je především v neschopnosti seřadit výsledek v pořadí (nejvíce relevantní dokumenty jsou nejvíce nahoře). Další nevýhoda je složitost pokládání dotazu, všechny termy v dotazu i v identifikaci dokumentu jsou chápány jako stejně důležité, relevantní dokumenty jsou pouze ty, které lexikálně obsahují zadané pojmy. Pro některé dotazy dostaneme úplně špatné dokumenty. Není totiž jasné, zda hledaný dokument má obsahovat současně všechny nebo jen některé pojmy (synonyma). [19]

V této práci booleovský model zatím nevyužívám. Pokud bych tuto práci rozšířil, mohl bych booleovský model použít pro vyhledání stránek, které obsahují jen určité návrhové vzory. Metodou (popsanou dále v textu) získám návrhové vzory. Automaticky (programem RWV) určím, zda se návrhový vzor na stránce vyskytuje (ohodnocení ano/ne). Pomocí booleovského dotazu získám stránky, které obsahují požadované návrhové vzory. Př. titulek AND nabídka AND login AND NOT teplota.

3.2 Vektorový model

Vektorový model vznikl v 70. letech. Snahou bylo odstranit problémy Booleovských DIS. Hlavním rozdílem oproti předchozímu modelu je reprezentace dokumentů a

uživatelských dotazů pomocí vektorů. Model obsahuje databázi D obsahující n dokumentů. $D = (d_1, \ldots, d_n)$. Dokumenty jsou popisovány pomocí m termů. $T = (t_1, \ldots, t_m)$. Každý dokument je reprezentován pomocí vektoru vah $\vec{d_i} = (w_{i,1}, \ldots, w_{i,m}) \in < 0, 1 >^m$. Pro vektor dotazu Q platí stejná pravidla.

- Pokud $w_{i,j} = 0$, znamená to, že term t_j není pro identifikaci dokumentu d_i vůbec důležitý.
- Pokud $w_{i,j} = 1$, znamená to, že term t_j je pro identifikace dokumentu d_i velice důležitý.

Indexový soubor vektorového modelu potom představuje matici:

$$D = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{pmatrix} \in < 0, 1 >^{n \times m}$$

Ohodnocení slov ve vektorovém prostoru probíhá následovně. Hodnocení slov je funkce, která každému slovu přiřadí nějakou číselnou hodnotu. Existuje několik typů ohodnocení slov.

1. Binární - pokud se slovo v dokumentu vyskytuje alespoň jedenkrát, bude slovo ohodnoceno číslem 1, jinak 0. Je to velmi ztrátová reprezentace, ale v mnoha algoritmech poskytuje dobré výsledky. Více v [14].
2. Frekvenční (TF) - u každého slova spočítá jeho frekvenci. Frekvence klíčového slova v dokumentu je číslo, které udává počet výskytů klíčového slova v dokumentu. Někdy se používá normalizovaná frekvence klíčového slova. Frekvence je normalizovaná délkou dokumentu. V této práci využívám tuto metodu pro ohodnocení.
3. TFIDF(Term Frequency Inverse Document Frequency) - Jedna z nejpoužívanějších metod. Tato metoda se snaží zohlednit frekvenci slova vzhledem k jeho výskytu v ostatních dokumentech. Čím častěji se slovo vyskytuje v ostatních dokumentech, tím méně důležité bude.[8].

   $$TFIDF = TF_i \times log \frac{D}{DF_i}$$

   TF_i je počet výskytů termu T_i v dokumentu. D je počet všech dokumentů. DF_i je počet dokumentů neobsahující term T_i.

Hlavním problémem vektorového modelu je veliká dimenze. Z hlediska výše uvedených metod je problematické odvozovat závěry ze stovek (tisíců) příkladů, kde každý příklad obsahuje desítky až stovky atributů. Jednou z možností je použít metody redukce dimensionality nebo použít jen určité termíny. Pro selekci atributů se dá použít χ^2 (chi-square test). Pro transformaci termínu lze použít shlukování, faktorovou analýzu, nebo indexaci latentní sémantiky (LSI). Pro snížení dimenze prostoru se používá metoda Singulární rozklad (SVD). Ta nám umožňuje provést redukci dimenze prostoru dokumentu při zachování shluků. Více o této metodě napsal Michal Krátký v [14].

V předchozím textu jsem popsal všechny náležitosti vektorového modelu a zbývá popsat jak se provede výpočet podobnosti mezi dotazem Q a dokumentem D. V praxi se používá velké množství měr podobností, které vyjadřují vztahy mezi dokumenty. Výběr míry podobnosti záleží na jakých datech chceme provádět výpočet. Ja používám data intervalová, pro které se obvykle používá míra Kosinová. Touto mírou se spočítá úhel, který svírá dokument D s dotazem Q. Viz vzorec (1). Další míry: Jaccradova viz. vzorec (2), Diceova míra viz. vzorec (3).

$$Sim(Q,D) = \frac{\sum_{i=1}^{n}(D_i \cdot Q_i)}{\sqrt{\sum_{i=1}^{n}(D_i)^2 \cdot \sum_{i=1}^{n}(Q_i)^2}} \tag{1}$$

$$Sim(Q,D) = \frac{\sum_{j=1}^{m} Q_j * D_j}{\sum_{j=1}^{m} Q_j + \sum_{j=1}^{m} D_j - \sum_{j=1}^{m} Q_j * D_j} \tag{2}$$

$$Sim(Q,D) = \frac{2 * \sum_{j=1}^{m} Q_j * D_j}{\sum_{j=1}^{m} Q_j + \sum_{j=1}^{m} D_j} \tag{3}$$

Více o vektorovém modelu v [22] nebo zde [19].

4 Mining

4.1 Data Mining

Data Mining (česky „dolování dat“) je proces hledání informací a znalostí ve velkém objemu dat. Jedná se o nástroj používaný v oblasti analýzy dat, sloužící jako podklady pro manažerské činnosti. Pro Data Miming se využívají technologie rozpoznání vzorů, statistické a matematické metody.

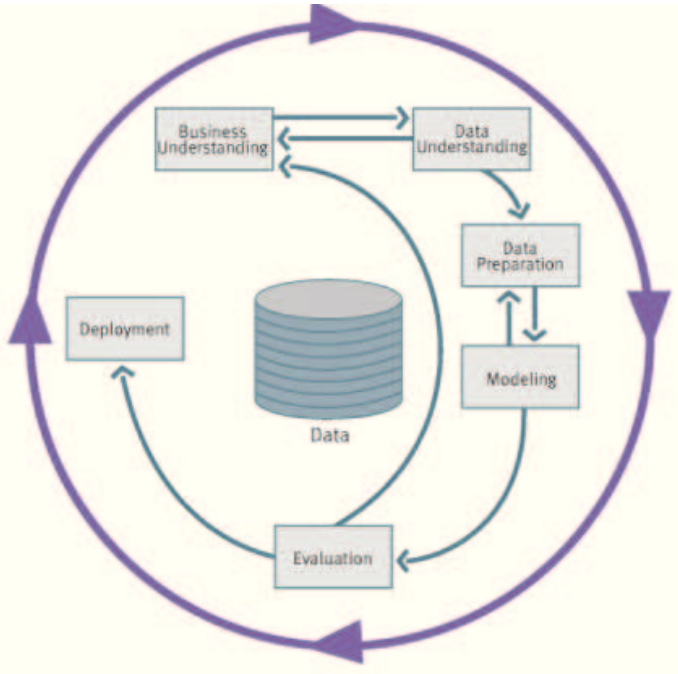
První náznaky se objevily v 60. letech. Rozvoj statistických metod, databázových aplikací a umělé inteligence vedl k prvnímu využití data miningu v praxi. Mohl nastat problém s hledáním vzájemných vztahů ve velkých datových souborech. Ve velkém souboru může vzniknout nepravidelná změna v datech bez možnosti vytvoření pravidla a bez praktického využití. V 90. letech byly vybudovány metody, umožňující vyhnout se předchozímu problému. Rostla poptávka po metodách zpracování rozsáhlých databází, které obsahují velké objemy dat. Z nich je obtížné získat potřebná data pomocí klasických tabulkových metod. To napomohlo k rychlému rozvoji Data Miningu a jeho rozšíření do komerční praxe. Existuje široká nabídka specializovaných softwaru pro tento účel (např. SAS Enterprise Miner, SPSS Clementine, STATISTICA Data Miner, Weka, Orange).

Definice 4.1 *Data Mining je proces výběru, prohledávání a modelování ve velkých objemech dat, sloužící k odhalení dříve neznámých vztahů mezi daty za účelem získání obchodní výhody.*

Usama M. Fayyad

Dolování dat je extrakce zajímavých netriviálních, potencionálně užitečných, skrytých souvislostí(vzory, informace) ve velkých objemech dat. DM slouží k porozumění dříve neznámých vztahů mezi daty a tvoření pravidel chování, které se dlouhodobě opakují. Některé databáze jsou velice objemné, a tudíž orientace v nich je prakticky nemožná (samotný tvůrce databáze neví jaká data obsahují). Pro takovou databázi by bylo přínosné použít techniku, která by dokázala z těchto rozsáhlých dat „vytěžit“ důležité informace nebo vzory chování. Více v [2], [3], [5].

Vzor v Data minigu chápu jako vztah mezi daty v databázi. Vysvětlím na příkladu: Zákazník nakupuje přes internetový obchod. Koupí si nový digitální fotoaparát a zároveň s ním si koupí paměťovou kartu. Takovou činnost provede polovina lidí, kteří si kupují fotoaparát. Taková činnost se dá zobecnit do vzoru. Vzor popisuje pravidlo, že pokud si zákazník koupí fotoaparát je pravděpodobné, že si koupí i paměťovou kartu. Tohoto faktu se dá využít při tvorbě internetového obchodu (u fotoaparátů umístím i paměťové karty).



Obrázek 3: Životní cyklus Data Miningu

Zdroje dat
V následujícím seznamu uvedu přehled různých zdrojů dat. Uvedeno v [4].

- Databáze, datové sklady
- Datové toky
- Strukturovaná data, grafy, sociální sítě.
- World Wide Web

4.1.1 Životní cyklus Data Mining

Vznik standardizovaného metodologického postupu. CRoss-Industry Standard Proces for Data Mining (CRISP-DM)[6], je to standardizovaný proces pro všechny zdroje dat. Metodologie je vždy stejná a popisuje data mining v následujících šesti krocích. Jejich návaznost ukáže obrázek 3, který je převzatý z [6]. Jednotlivé procesy popíšu.

- **Porozumění problematice (Business Understanding)**. „Tato počáteční fáze se zaměřuje na pochopení cílů projektu a požadavků z manažerského hlediska a

poté převedení těchto poznatků do definování problému data miningu. V této fázi dochází také k návrhu a tvorbě plánu pro řešení daného problému.“ Citováno z [21].

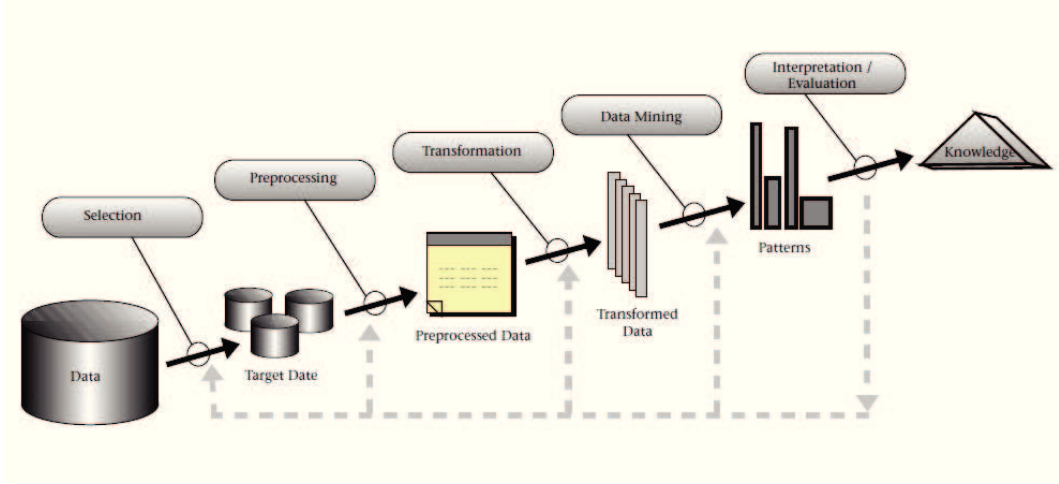
- **Porozumění datům (Data Understanding)**. „Fáze porozumění datům začíná prvotním sběrem dat a pokračuje aktivitami vedoucími k seznámení se s daty, určením kvality dat, prvním nahlédnutím do dat nebo odhalením zajímavých podmnožin k vytváření hypotéz pro skryté informace. Tyto hypotézy se v průběhu celého procesu snažíme potvrdit. Někdy však můžeme hypotézu vyvrátit nebo naopak najít jiné řešení.“ Citováno z [21].

- **Příprava dat (Data Preparation)**. „Fáze přípravy dat zahrnuje všechny aktivity na vybudování konečného datového souboru, který bude zpracováván jednotlivými analytickými metodami. Tato data by tedy měla obsahovat údaje relevantní k dané úloze a mít podobu, která je vyžadována vlastními analytickými algoritmy. Tento proces nelze správně provést bez znalosti dat. Špatná integrace dat by mohla vést ke znehodnocení zdrojů dat a ovlivnění celkové kvality řešení.“ Citováno z [21].

- **Modelování (Modeling)**. „V této fázi jsou zvoleny a aplikovány různé modelovací techniky a jejich parametry jsou kalibrované na optimální hodnoty. Obvykle existuje řada různých metod pro řešení dané úlohy (doporučuje se použít více různých metod a vybrat několik nejlepších, které postupují do dalšího kroku) a vhodně nastavit jejich parametry. Některé techniky mají specifické požadavky na podobu dat. Pak je tedy často potřeba přistoupit zpět k fázi přípravy dat.“ Citováno z [21].

- **Vyhodnocení (Evaluation)**. „V této fázi máme sestavený model (nebo modely), který se zdá mít z hlediska analýzy dat vysokou kvalitu. Před konečným využitím modelu je důležité důkladněji vyhodnotit model a přezkoumat kroky vedoucí ke stavbě modelu k nabytí jistoty, že skutečně dosáhneme daných cílů. Hlavním cílem je zjistit, zda existuje nějaká důležitá záležitost, která nebyla dostatečně zahrnuta. Na konci této fáze by mělo být dosaženo rozhodnutí o využití výsledků data miningu. Dle získaných výsledků je již možno zvážit případnou implementaci celého procesu.“ Citováno z [21].

- **Využití výsledků (Deployment)**. „Je posledním krokem v celém procesu. Je však nutné podotknout, že proces nekončí, ale začíná se cyklicky opakovat. Pokud se zákazník rozhodne výsledky data miningu implementovat do svých procesů, je nezbytné modely udržovat aktuální. Závislosti v datech se časem mění, a pokud by systém nebyl dostatečně robustní či pravidelně aktualizován, je velmi



Obrázek 4: Pozice Data Miningu v procesu získávání znalostí z databází

pravděpodobné, že by časem pozbyl kvality, tak i zcela své funkce. Proto je nutné pravidelně ověřovat funkci modelu novými daty a tím udržovat aktuálnost modelů. Vytvoření modelu obecně není konec projektu. Získané znalosti budou muset být zorganizovány a prezentovány způsobem, aby je zákazník mohl využít. V závislosti na úkolu může být tato fáze zcela prostá – pouhé sepsání závěrečné zprávy, nebo také složitá – zavedení systému pro automatickou klasifikaci nových případů.“ Citováno z [21].

Obrázek 4 je převzatý z publikace [5]. Na obrázku 4 je zobrazena pozice Data Minigu v procesu získávání(dolování) informací z databáze. V prvním kroku se provede selekce dat. Pokud jsou data pro analýzu relevantní jsou načtena z databáze. V následujícím kroku se upraví data. Zde dochází k integraci více datových zdrojů, čištění a úpravě dat do podoby, kterou vyžadují analytické nástroje a metody. Odstraní se šum a nekonzistentní data. V datech se můžou vyskytnout různé chyby (zákazník nakoupil zboží před tím než se narodil), tyto chyby je potřeba odhalit a opravit. V kroku 3 jsou data transformována nebo sloučena do forem vhodných pro dolování. Například se provede souhrn nebo agregace. Někdy jsou procesy transformace a konsolidace dat aplikovány do procesu výběru dat, zvláště pak v případě datových skladů. Někdy je redukce dat provedena za účelem získat menší zastoupení původních dat bez ztráty integrity. Krok 4 je základní proces, kde se získávají datové vzory za použití inteligentních metod. V posledním kroku se určí vzory. Uživateli se vizuálně prezentují data získané dolováním. Je vytvořen model, který musí přinést užitek. Příklad úlohy pro data mining: Kolik bude stát poze-

mek 50km od Ostravy? Které produkty se prodávají společně? Které produkty si lidé kupují společně s jinými? [4]

4.1.2 Kategorie metod

Data mining se dá použít na řešení mnoha různých problémů. Podle druhu problému můžeme vytvořit určité skupiny, které mají za cíl postihnout neznámé vztahy mezi daty. Jednotná podoba dělení není definována. Můžeme se setkat s různými skupinami. Já jsem vybral dělení podle Usamy Fayyada [5]. Fayyad určuje dva hlavní cíle data miningu, predikci a deskripci.

- **Predikce** Umožňuje předpovídat budoucí hodnoty atributů na základě nalezených vzorů v datech. Tyto metody se dají využít například pro předpověď počasí, vývoj ceny na burze a mnoho dalších.
- **Deskripce** Je brána jako samozřejmost. Pokud chcete někomu předat nějaké informace, musíte být schopni danou skutečnost popsat. Deskripce popisuje nalezené vzory a vztahy v datech, které mohou ovlivnit rozhodování.

Cílů predikce a deskripce je dosaženo pomocí následujících úkolů:

- **Klasifikace**. Klasifikace rozděluje objekty podle jejich charakteristických rysů do jednotlivých klasifikačních tříd. Tato třída (můžeme nazývat doména) je vytvořená předem z množiny trénovacích dat. Každý objekt můžeme někam zařadit. (Např.: stránky internetového obchodu můžeme určitě zařadit mezi třídu *nákup*)
- **Regrese**. Data která jsou obsažena v databázi mají informativní hodnotu, ze které můžeme předpovídat jaké další hodnoty budou následovat.
- **Shlukování**. Shlukování je skupina metod, které tvoří shluky dat. Vstupní množina dat je rozdělena některou z technik do shluků (počet takových skupin je znám buď předem, nebo až při průběhu shlukování). Používanými technikami jsou rozhodovací stromy, neuronové sítě, logistická regrese, diskriminační analýza.
- **Sumarizace**. Zahrnuje metody pro hledání uceleného popisu podmnožiny dat - podává přehled o struktuře dat. Některé metody zahrnují odvození pravidel z vícerozměrných zobrazovacích metod a objevení funkčních vztahů mezi nimi. Jsou aplikovány na průzkumné analýzy dat a automatické vytváření zpráv.

- **Modelování závislostí**. Hledá model, který popisuje důležité závislosti mezi proměnnými. Rozdělujeme jej na dvě úrovně: 1. Strukturální úroveň modelu (specifikuje proměnné, které jsou na sobě logický závislé, často graficky). 2. Kvantitativní úroveň modelu specifikuje síly závislostí za použití číselné stupnice.
- **Detekce změn a odchylek**. Kontroluje nejpodstatnějších změny v datech od původně naměřených nebo normativních hodnot.

4.2 Text Mining

V posledních letech lze vidět ohromný nárůst množství dokumentů dostupných na Internetu, nebo např. v podnikových intranetech. S rostoucím počtem dat vzrůstá potřeba po kvalitních metodách, které vyhledávají a zpracovávají texty. Textová data jsou obvykle uložena v nestrukturované podobě (obyčejný text v článku). Data obvykle nejsou uložena v databázích, ale především na webových serverech, souborových systémech nebo na PC. Příklad dat pro Text Mining:

- elektronická pošta
- internetové dokumenty (poznámky, prezentace, zápisky)
- technické zprávy
- informační kanály

Vyhledáváním a zpracováním textů se zabývá Text Mining(TM). TM je jedna z úloh obecného Data Miningu(DM), který jsem popsal v kapitole 4.1. V dnešním světě internetu je většina (téměř 80%) všech informací uložena v podobě textových dokumentů (pouhých 20% je strukturovaně uloženo v databázích). Z důvodu rozdílnosti dat se oddělilo dolování v textech od dolování v datech. Bylo potřeba vytvořit nové metody předzpracování a zpracování textů. U TM usilujeme o extrakci zajímavých vzorů z textových dokumentů. Nejdůležitější úlohy Text Miningu jsou:

- **Kategorizace**
- **Shlukování**
- **Extrakce informací**
- **Sumarizace**

„Dolování v textech lze definovat jako proces objevování (získávání) znalostí, který má za cíl identifikovat a analyzovat užitečné informace v textech, jež jsou zajímavé pro uživatele. Dolování v textech lze také definovat jako netriviální extrakci implicitních, předem neznámých a potencionálně užitečných informací z (velkého množství) textových dat. Předem neznámými informacemi jsou myšleny informace, které znal autor dokumentu, a které nejsou implicitně viditelné. Nalezení zcela nových informací je velmi obtížný úkol, který se často realizuje v souborech textů, kde se analyzují vzájemné vazby a souvislosti.“ Jak je pěkně uvedeno v [7].

Text Mining můžeme rozdělit na dvě části:

1. **Předzpracování textu** - příprava textu do formy, se kterou se provádí další zpracování.

2. **Získávání znalostí - vzorů** - v této části se odvozují vzory z před připraveného textu. Dochází k analýze vygenerovaných termů a k rozhodovacímu procesu vedoucímu k poskytnutí požadovaných výsledků (zařazení dokumentu do kategorie).

4.2.1 Předzpracování textu

Důležitým krokem, který předchází dolování dat, je příprava těchto dat. Při předzpracování je vstupní dokument převáděn do určité podoby - do odpovídajícího formátu. S touto podobou se provádí další zpracování. Ze vstupních dat je extrahován pouze text. Jsou odstraněny obrázky nebo informace, které se netýkají textu. Text je převeden na stejný druh písma. Co je zachováno je textová struktura textu. Tato struktura může pomoci k určení významu termů. Term je ustálený celek v dokumentu(základní objekt, s nimž se provádí další zpracování). Term se skládá z jednotlivých slov, nebo více frází spolu s určením slovního druhu. Výběr termu může být ruční, nebo automatizovaný. Dále jsou z textu vyjmuta slova, která nenesou žádný význam. Vektor termů může být velice velký, a proto se aplikují metody, které umožňují snížit velikost na přijatelnou mez. [9],[11],[10], [7].

Metody předzpracování textu

- **Převod na malá nebo velká písmena:** Všechny znaky v textu jsou převedeny na malá, nebo velká písmena. Ve svojí práci text převádím na malá písmena. Tímto se zbavím rozdílu mezi slovem, které je napsané velkým písmem respektive malým. [13].

- **Odstranění diakritiky:** Všechny znaky v textu projdou odstraněním interpunkčních znamének. Analýza textu se tím zase o něco zlepší. [13].
- **Stemming:** Převod slova do základního tvaru je velmi často používaná technika [13]. Smyslem stemmingu je sjednocení slov se stejným významem, ale jiným tvarem. Jako příklad můžou sloužit anglické tvary slov *looks, looking a looked*, které po úpravě mají tvar *look*. V češtině například slovo *program* má tvary *programování* nebo *programy*. Tato technika dokáže snížit výslednou velikost vektoru. Pro anglicky psané texty se používá algoritmus[8], který byl napsán Martinem F. Porterem v roce 1979. Tento algoritmus využívá pouze odstraňování přípon. U předpon se předpokládá, že jich v angličtině není tolik a nejsou tak často použity. Pro češtinu se používá například morfologický analyzátor Ajka více v[12].
- **Stop list:** Je to seznam zakázaných slov, která nechceme zahrnout do analýzy. Používá se především pro odstranění slov, která nemají žádný význam. Jsou to například *a, nebo, než, aniž, třeba* a mnoho dalších. Tyto slova jsou manuálně nashromážděná. Jiná možnost než stop list je vypustit určitý počet (např. 100, 200, 300) nejfrekventovanějších slov v daném jazyce. Dále je nutné odstranit slova, která jsou obsažena ve velkém množství dokumentů (pak je význam těchto slov nulový).
- **Odstranění dalších informací:** Takové informace obvykle nese www dokument. Jsou to HTML značky, skripty a jiné. Tyto značky nechceme do analýzy začlenit.
- **Ohodnocení slov:** Nejpoužívanějším způsobem reprezentace dokumentu je použití vektoru, který má tolik složek, kolik je slov v souboru dokumentů. Jednotlivé dokumenty bývají reprezentovány řídkými vektory o tisících hodnot. Více o ohodnocení jsem zmínil v kapitole 3.2.

4.2.2 Klasifikace dokumentů

Klasifikace dokumentů je proces, sloužící k vytvoření modelu, pro přiřazování nebo rozlišování objektů do předem známých tříd na základě jejich vlastností. V procesu Text Mining se kategorizace používá pro hledání správných témat pro každý dokument. Dokument může být zařazen podle obsahu, názvu, autora apod. Každý dokument může být v několika, jedné nebo žádné třídě. Kategorizace se uplatní v aplikacích indexování, řazení, filtrování či organizování textu nebo třídění webových stránek nebo odhalování spamu. Kategorizace dokumentu je používaná již od 60. let minulého století. Až do konce

[8] http://tartarus.org/ martin/PorterStemmer/index-old.html

80. let se klasifikační pravidla vytvářela ručně. Poté byly vyvinuty systémy, které na základě již zařazených dokumentů dokázaly tyto pravidla vytvořit automaticky.

Klasifikace textových dokumentů patří do problému strojového učení. Cílem je automaticky přiřadit textový dokument do nějaké kategorie v závislosti na jeho obsahu. Pro kategorizaci lze použít obecně známé problémy jako jsou rozhodovací stromy, induktivní logické programování (ILP), asociační pravidla, Bayesova klasifikace, k-NN a jiné.

Pro klasifikační algoritmy, je třeba textový dokument reprezentovat jako vektor, kde každá složka reprezentuje frekvenci slova či termu. Klasifikaci můžeme rozdělit do dvou fází. V první fázi na základě trénovacích vzorů (u nichž víme, do jaké skupiny patří) určíme pravidla, podle nichž bude klasifikace prováděna. Ve druhé fázi jsou pravidla získaná v předchozím kroku testována na jiných vzorech, následně použita pro zařazování nových dat. Více o klasifikaci dokumentů například zde: [15].

4.2.3 Shlukování

Úkolem shlukování je nalézt shluky tak, aby platilo: dokumenty uvnitř shluku jsou si nejvíce podobné, a aby jejich podobnost s dokumenty z jiných shluků byla menší. Pomocí shlukování můžeme zjistit témata, která se vyskytují ve zkoumané množině dokumentů. Na rozdíl od klasifikace neznáme jednotlivé skupiny. [16],[11].

Kvalita shlukování je úměrná schopnostem porovnat odlišnosti jednotlivých objektů. Je důležité vybrat vhodné vstupní parametry. Vychází se z toho, že umíme měřit vzdálenosti mezi objekty. Následuje definice shluku.

Definice 4.2 *Mějme množinu objektů $O = O_1, \ldots, O_n$ a míru vzdálenosti objektů V. Shlukem nazveme takovou podmnožinu $X \in O$, pro niž platí*

$$max(V(O_i, O_j)) < min(V(O_k, O_i)),\ O_i, O_j \in X,\ O_k \notin X$$

Kapitolu o shlukování jsem uvedl, protože bych ho v budoucnu mohl použít pro shlukování webových stránek. Pokud bych kolekci různých webových stránek popsal pomocí vektorového modelu. U každého dokumentu provedu ohodnocení termů metodou (TFIDF). Pak můžu spustit shlukování. Ideálním výsledkem bude nějaký počet shluků, kde každý shluk odpovídá jedné doméně. Pro shlukování bych vybral algoritmus (Hierarchický), který množinu stránek postupně rozdělí do *k* shluků.

Následující vzorce a text pro výpočet vzdálenosti jsem si vypůjčil z [17]. „Předpokládejme, že každý příklad je charakterizován m numerickými veličinami. Vzdálenost mezi dvěma příklady $x_1 = [x_{11}, \ldots, x_{1m}]$ a $x_2 = [x_{21}, \ldots, x_{2m}]$ lze vyjádřit různými mírami. Např.:

- Hammingova vzdálenost

$$d_H(x_1, x_2) = \sum_{j=1}^{m} |x_{1j} - x_{2j}|$$

- Euklidovskou vzdálenost

$$d_E(x_1, x_2) = \sqrt{\sum_{j=1}^{m} (x_{1j} - x_{2j})^2}$$

- Čebyšova vzdálenost

$$d_C(x_1, x_2) = max_j |x_{1j} - x_{2j}|$$

Ve všech výše uvedených případech se jedná o speciální příklady Minkovského metriky

$$L^{(z)}(x_1, x_2) = \sqrt[z]{\sum_{j=1}^{m} (x_{1j} - x_{2j})^z}$$

$$d_H(x_1, x_2) = L^{(1)}(x_1, x_2), \quad d_E(x_1, x_2) = L^{(2)}(x_1, x_2), \quad d_C(x_1, x_2) = \lim_{z \to \infty} L^{(z)}(x_1, x_2)$$

Výše uvedené míry vzdálenosti závisí na měřítku veličin. Proto je třeba veličiny normovat. Konkrétní hodnota se obvykle dělí nějakou jinou hodnotou: průměrem, směrodatnou odchylkou nebo rozpětím (max-min). Navíc předpokládají stejný rozptyl u všech veličin. V případě různého rozptylu se doporučuje použít Mahalanobisovu vzdálenost, která je zobecněním vzdálenosti euklidovské.“ Citováno z [17].

$$d_{M^2}(x_1, x_2) = (x_1 - x_2)^T S^{-1} (x_1 - x_2)$$

Na závěr uvedu jeden z nejpoužívanějších vzorců pro měření podobnosti dokumentů - Kosinová míra.

$$Sim(x_1, x_2) = \frac{\sum_{i=1}^{n} (x_{1i} \cdot x_{2i})}{\sqrt{\sum_{i=1}^{n} (x_{1i})^2 \cdot \sum_{i=1}^{n} (x_{2i})^2}}$$

4.2.4 Metody shlukování

Shlukování nespadá do jednoho oboru, ale spadá do více oblastí, je to například matematika, statistika, numerická analýza či umělá inteligence. Následuje přehled metod pro shlukování převzatý z [18].

- Hierarchické Metody

  - Aglomerativní algoritmy
  - Rozdělovací algoritmy
- Rozdělovací metody
  - Pravděpodobnostní shlukování
  - k-medoids metody
  - k-means metody
- Algoritmy založené na hustotě prvků
- Metody založené na mřížkách
- Jiné metody (Např.: metody pro mnohorozměrná data, různé metody založené na strojovém učení nebo metody využívající neuronové sítě).

4.3 Web Mining

Web mining je technika, která využívá metod Data Miningu, pro objevování vzorů na webových stránkách (hledá a analyzuje užitečné informace). Na Web Mining by se dalo pohlížet jako rozšíření Data Minigu s tím rozdílem, že se používá pro data z webu. Hlavním úkolem Web Miningu je klasifikace webového dokumentu. Web mining se dělí na tři skupiny, které objasním v následujících kapitolách [26], [27].

4.3.1 Web content mining

Jeden z cílů web content miningu je analyzovat webové stránky a zjistit, které užitečné informace jsou na nich obsaženy. Užitečné informace z pohledu uživatele (text, odkazy, obrázky, zvuk, video). Někdy je označován jako text mining, protože se většinou zabývá zpracováním textu. Více o použití v [27]. Web content mining se dá rozdělit na dvě oblasti:

- Information Retrieval - účelem je získat užitečné informace, pomocí kterých nalezneme relevantní webové stránky na internetu.
- Extrakce informací - účelem je zjistit informace o struktuře, které mohou být uloženy v databázi (např. název, cena a termín zájezdu).

Podle publikace [23] pro oblast web mining je vhodné použít webové návrhové vzory. Vzory se používají odlišným způsobem než obvykle (podle vzoru programátor naprogramuje požadovanou akci - stále se opakující). V publikaci [23] byla vybrána sada

návrhových vzorů, které se vyskytují na skutečných stránkách (tyto návrhové vzory jsou vnímány uživatelem). Návrhové vzory jsou pak použity pro popis charakteru webové stránky (na diskusním fóru se vyskytuje vzor přihlášení, příspěvky a vložení nového příspěvku). Pro tento účel musel být vytvořen katalog těchto vzorů[9]. Cílem použití návrhových vzorů je popis webové stránky a poté tento popis opakovaně používat pro jiné úkoly. Vlastnosti návrhového vzoru: je málo závislý na implementaci, je závislý na vnímání uživatele, prvky jedné konkrétní instance vzoru jsou na stránce pohromadě. Pro extrakci návrhového vzoru jsou použity tzv. Gestalt principy pro vizuální systémy. Byly vybrány 4 nejvhodnější pravidla založená na Gestalt principech. Jsou to:

1. Proximity (blízkost) - související informace bývají blízko u sebe.
2. Similarity (podobnost) - podobně vypadající prvky obsahují podobné informace
3. Continuity (souvislost) - informace následují plynule za sebou a doplňují se.
4. Closure (celek) - související informace bývají společně uzavřeny do celků.

Metodika mojí práce je obdobná jako metodika uvedená v práci [23].

4.3.2 Web structure mining

Web structure mining se zabývá strukturou hypertextových odkazů v rámci webu (struktura webu). Analýza hyperlinků je stará oblast výzkumu, ale s rostoucím zájmem o dolování dat z webu se zvýšilo úsilí a vyústilo v novou oblast tzv. Link Mining. Web obsahuje řadu objektů s téměř žádnou jednotící strukturou, rozdíly ve vývojovém stylu a obsahu jsou mnohem větší než v tradičních knihovnách, proto se analyzuje vzájemné propojení WWW stránek. Objekty WWW jsou webové stránky a odkazy. Úkoly link miningu:

- Link - klasifikace. Úkolem je zaměřit se na předpověď kategorie dokumentu, na základě slov které se objevují v dokumentu, vazby mezi stránkami, html tagy a další možné atributy.
- Link - shlukování. Cílem je najít přirozeně se vyskytující pod-třídy. Dokumenty se rozdělují do skupin, kde podobné dokumenty jsou pohromadě a odlišné v různých skupinách.
- Link - typ. Existuje široké spektrum úkolů týkajících se predikce existence vazeb (predikce typu spojení dvou dokumentů nebo účel spojení).

[9] http://www.welie.com/patterns/index.php

- Link - váha. Odkazy mohou být ohodnoceny - vážení odkazů.
- Link - mohutnost. Předpověď kolik vazeb je mezi dokumenty(transformace WWW do orientovaného grafu).

Cílem je vytvoření modelu organizace Webu. Pomáhá zdokonalovat vyhledávací roboty (page rank - důležitost stránky se zvyšuje tím víc, čím víc se na ni odkazuje jiných relevantních stránek) [24].

4.3.3 Web usage mining

Web usage mining se zaměřuje na využití techniky, která by mohla předpovídat chování uživatelů, zatímco používají WWW. Web usage mining je založen na sbírání dat z logovacích záznamů a z nich vytvoření nových vzorů chování uživatele. Chování uživatelů může objevit nové vazby a závislosti (odvození vazeb tam kde nebyly dosud navrženy nebo tam kde jsou chybné). Problémy týkající se web usage miningu:

1. Předzpracování - dostupné údaje obsahují šum, jsou nekonzistentní a neúplné. Čištění, integrace, transformace a redukce dat.
2. Objevení vzoru - několik různých metod a algoritmů (statistika, data mining, strojové učení a rozpoznání vzorů) by mohly být použity na identifikaci vzoru chování uživatele.
3. Analýza vzoru - pochopení, vizualizace a výklad.

S použitím web usage miningu souvisí ochrana osobních údajů. Více o usage miningu v [25].

5 Vlastní metoda řešení

5.1 Metodika sběru dat

Abych mohl vytvořit vlastní aplikaci, která dokáže automaticky detekovat vzory, musel jsem sesbírat informace o doménách. Takové informace jsem nalezl na stránkách, které se týkají stejného tématu. Stránky je nutné uložit a důkladně prozkoumat. Na doméně nezáleží, aplikace musí být schopna vyhledávat všechny druhy stránek. Pro sběr dat jsem použil aplikaci Internet Explorer. Pomocí internetového vyhledávače Google.cz jsem vyhledal jen české internetové stránky. Anglicky psané stránky mě pro tuto chvíli nezajímají. Pro anglicky psané stránky budou fungovat stejné algoritmy, ale jiné konfigurační soubory. Pro vytvoření konfiguračního souboru musím analyzovat co možná největší počet HTML souborů. Existují postupy pro automatické získávání těchto konfiguračních souborů (tzv. strojové učení). Aplikace se dokáže učit novým poznatkům, které nejsou explicitně definovány. Tento postup není příliš vhodný pro tento typ úlohy, protože většina automatických způsobů indexování je založena na pozorování, že významnost klíčových slov pro indexaci přímo souvisí s frekvencí výskytu klíčového slova (term) v dokumentu. Pro moje potřeby nepotřebuji frekvenci jednotlivých slov, ale frekvenci bloků sestavených ze slov a to vyžaduje lidský cit. Další nevýhoda strojového učení je velký objem trénovacích dat. Z tohoto důvodu se přikláním k ručnímu sestavení konfiguračních dat. Stránky, které budu analyzovat jsem vybral na základě velikosti relevance k dotazu. Všechny postupy pro sběr dat jsem popsal v několika krocích:

1. V prvním kroku jsem si vybral doménu, na které budu provádět pokusy. Konkrétně jsem si nevybral jen jednu, ale více. Jsou to tyto domény: Dovolená, Technické údaje, Diskusní fórum. Postup při sběru dat budu demonstrovat pouze na doméně dovolená. Pro ostatní domény se postup neliší.

2. Pro sběr trénovacích dat jsem si vybral aplikaci Internet Explorer verze 8. Spustím IE a zadám adresu http://www.google.cz. Jako klíčové slovo pro vyhledání jsem zadal ”dovolená”a stisknu tlačítko hledat.V tomto kroku mi google našel přibližně 12 000 000 výsledků za 0,06s. V dalším kroku zakliknu stránky pouze česky. V této fázi google našel přibližně 13 100 000 výsledků za 0,11s.

3. Ve třetím kroku začnu procházet seznam stránek, které google vyhledal a jsou seřazeny podle míry relevance k dotazu (v tomto případě je dotaz dovolená). Postupně otvírám každou stránku. Prohlédnu si ji, a pokud odpovídá dotazu, uložím si ji na disk.

4. V předchozím kroku jsem našel vyhovující www stránku. Tuto stránku uložím na disk klasickým způsobem. Při ukládání nastavuji dva parametry. První parametr je *Uložit jako typ*. Pro moje potřeby jsem zvolil uložení *Webová stránka, pouze HTML*, tedy každou stránku uložím tak, aby obsahovala html značky a všechen text. Nic víc pro potřeby analýzy nepotřebuji. Druhý parametr, který nastavuji je *kódování*. Zde si můžu vybrat z několika jazyků. Pro moje potřeby a potřeby aplikace si stránky ukládám do kódování *Středoevropské jazyky (Windows)*.

5. Postupně opakuji krok 3 a 4 tak dlouho, až budu mít uložených nejméně 20 stránek.

Tímto krokem sběr dat končí a přecházím ke kapitole 5.2, kde budu stránky ručně procházet a zaznamenávat si z mého pohledu užitečné informace.

5.2 Metodika analýzy dat

V kapitole 5.1 jsem popsal postup při sběru dat. Tímto mám shromážděna potřebná data a na nich provedu analýzu. Analýzou rozumím ruční procházení jednotlivých souborů a zaznamenávání užitečných informací. Taková informace může být:

1. Titulek stránky - v titulku se nachází informace o stránce (např. dovolená, zájezd atd.) tato informace mi pomůže úspěšně detekovat stránku o dovolené.

2. Frekvence slov - frekvence slov je velice důležitá. Napomáhá nám rozeznat jaké termy jsou užitečné pro analýzu respektive neužitečné. Termy, které se na stránkách vyskytují často, jsou užitečné - vyskytují se v určité doméně s velkou frekvencí a tak popisují tuto doménu. Na druhé straně jsou termy, které se nevyskytují příliš často a tedy nemají pro mě žádný užitečný význam. Někde uprostřed jsou termy, které se nevyskytují příliš často, ale jejich výskyt není zanedbatelný. Tyto termy jsou nejvíce problematické. Tento problematický term může zvýšit relevanci nebo naopak snížit. To zjistíme až při testování aplikace v praxi. Problematiku výběru klíčových slov (dále jen KS) přiblížím na příkladu.

   Existují čtyři dokumenty, každý dokument obsahuje určitá KS. Dokumenty také můžu chápat jako úsek textu (tzv. segment). Z tabulky 1 je patrné, že KS a je obsaženo ve všech dokumentech a tedy je z pohledu rozlišení dokumentů bezvýznamné. Ostatní KS nejsou obsažena ve všech dokumentech, a proto nám přibližují resp. oddalují dokumenty. KS b a d obsahují dokumenty D1 a D2 a tedy jsou si podobné. Obdobně KS f a c obsahují dokumenty D3 a D4. Pokud takto zvolíme klíčová slova a následně položíme dotaz nad množinou dokumentu D1-D4. Výsledkem bude skupina S1 respektive S2. Viz tabulka 2.

Dokument	Klíčová slova
D1	a,b,d
D2	a,b,r,d
D3	a,c,f
D4	a,f,c,z

Tabulka 1: Klíčová slova

Skupina	Dokumenty	Klíčová slova dotazu
S1	D1,D2	b,d
S2	D3,D4	c,f

Tabulka 2: Skupiny dokumentů

3. Návrhové vzory vyskytující se na stránce - na pohled rozdílné, vykonávají stejnou činnost. Návrhový vzor je ukrytý v bloku textu. Tento text nazvu jako segment. Na stránkách si všímám segmentů, o kterých si myslím že by se mohly vyskytovat hlavně na stránkách o daném tématu. Tyto segmenty textu se často opakují na dané doméně. Tento text je v nějaké grafické podobě například tabulka nebo seznam. Jakmile zaznamenám segmenty a jejich frekvenci (označím Sf) ze všech stránek z kolekce "Dovolená", mohu vybrat klíčové segmenty. Pro každý segment stanovím jeho IDF (inverzní frekvence segmentu) podle vzorce $IDF(segment) = m/Fs$, kde m je celkový počet dokumentů v kolekci a Fs je počet dokumentů, ve kterých se vyskytuje segment. Protože kolekce dokumentů obsahuje jen kladné stránky, budu inverzní funkci chápat opačným způsobem. Pokud se na všech stránkách bude vyskytovat segment, bude se IDF rovnat $log(1) = 0$. V takovém případě je segment jasným favoritem. V opačném případě, pokud se segment vyskytuje jen na jedné stránce $log(20) = 1,3$, nebude pro mě důležitý. Vyberu takové segmenty, pro které platí *IDF* $<$ *hladina významnosti*.

   Tuto problematiku přiblížím na příkladu z domény dovolená. Na těchto stránkách se často vyskytuje vzor *Nabídka dovolené*. Je to úsek textu, kde se vyskytují slova: *destinace (stát), termín dovolené (datum), délka dovolené (počet dní), cena*. Tyto slova jsou blízko u sebe. Těchto úseků (návrhových vzorů) se na stránce vyskytuje obvykle více. Zaznamenáváním si vytvořím seznam návrhových vzorů, které na stránkách budu detekovat. U každého návrhového vzoru si zaznamenávám klíčová slova. Při výběru KS používám techniku popsanou v bodě 3 s tím rozdílem, že nerozlišuji

dokumenty, ale úseky (bloky) textu. Následující příklad popisuje návrhový vzor *Nabídka dovolené*. Pro jednoduchost jsem uvedl jen tři bloky. Data jsem převzal ze stránky *Bezva-dovolena.htm*, kde se tento blok opakuje 33 krát. Viz tabulka 3.

Blok	**Text**
B1	termín, 14.12.-21.12.2010, 7 dní, Strava, Polopenze, Cena, 11 130,-Kč
B2	termín, 18.10.-26.10.2011, 8 dní, Strava, Snídaně, Cena, 33 990,-Kč, (+2500 Kč taxy),konečná, cena, vč. poplatků, 36 490,-Kč
B3	18.01.-25.01.2011 - 7 dní, Strava, All Inclusive, Cena, 15670,-Kč

Tabulka 3: Text v bloku

Analýzou těchto klíčových slov jsem vytvořil množinu popisující návrhový vzor *Nabídka dovolené*. Viz tabulka 4.

Blok	**Klíčová slova**
B1	<datum>, <malé číslo>, <cena>, dní, polopenze
B2	<datum>, <malé číslo>, <cena>, dní, snídaně
B3	<datum>, <malé číslo>, <cena>, dnů

Tabulka 4: Klíčová slova v bloku

Popis tabulky 4:

KS <datum> je datový typ, který reprezentuje všechny datumy (např. 14.12., 18.10.-26.10.2011, 13. ledna 2000).

KS <malé číslo> je datový typ, který reprezentuje všechny čísla, které obsahují maximálně dvě cifry (např. 2, 5, 76).

KS <cena> je datový typ, který reprezentuje všechny ceny (např. 11130,-Kč, 15670,- , 11000Kč).

Sjednocením klíčových slov z bloku B1-B3 nám vznikne množina obsahující tato slova: datum, malé číslo, cena, dní, polopenze, snídaně. Tato množina nám reprezentuje návrhový vzor *Nabídka dovolené*. Jak je patrné KS *All Inclusive* se v množině nevyskytuje, protože se na stránce vyskytuje i na jiných místech. Jeho zařazení mezi KS by v některých případech znamenalo přiblížení bloků textu, které nemají nic společného. Bližší popis všech datových typů bude uveden v kapitole 6.

Poznámka: v praxi se nejedná přímo o klíčová slova, ale o jejich kmeny. Například u slov *dnů* a *dní*. Po odtržení koncovky *ů* od slova *dnů* nám vznikne kmen *dn* a

stejně tak u slova *dní* po odtržení koncovky*í* nám vznikne kmen *dn*. Tedy jsou stejné. Koncovkou je to, co se mění při skloňování nebo časování. (Příklad: ve slově *příspěvk/ů*, *příspěvk/y* je společný kmen *příspěvk*). Tento jednoduchý způsob je překvapivě úspěšný.

4. TAGy - v bodě 3 jsem získal slova, která popisují návrhový vzor. Návrhový vzor se obvykle vyskytuje v určitém místě stránky a má nějakou grafickou podobu, která je znázorněna uživateli. Tuto pozici (místo) popisuje HTML TAG. Příklad takového TAGu může být: *table (tabulka)*, *title (titulek)*, *tr (řádek tabulky)*, *li (položka v seznamu)*, *div (oddíl)* nebo *p (odstavec)*. Těchto HTML TAGů je mnohem více. Přehledný seznam všech je na internetových stránkách [10]. U každého segmentu si zaznamenám v jakém TAGu se vyskytuje. Tyto údaje využiji pro hledání návrhového vzoru. Tento nápad vysvětlím. Při vyhledání návrhového vzoru nebudu procházet celou stránku (ta je sestavena z HTML TAGů), ale budu procházet jen ty TAGy, ve kterých se vyskytoval návrhový vzor.

   Ručním zaznamenáváním jsem získal data, podle kterých bude aplikace vyhledávat. Uložení a popis těchto dat bude v kapitole 5.3.

5.3 Nastavení automatické detekce

Nastavení zapisuji do souboru XML. Soubor může obsahovat libovolný počet vzorů. Jeden XML dokument popisuje jednu doménu. Postupně budu vysvětlovat jednotlivé parametry. Jako příklad jsem zvolil doménu dovolená a návrhový vzor (segment) Nabídka dovolené. Ve výpisu 1 neuvádím všechny parametry. Kompletní výpis uvedu v příloze B.

```
<vzor>
  <jmeno>Dovolená</jmeno>
  <segment id = "nabidka">
    <seg_hodnota>4</seg_hodnota>
    <seg_maxVelikost>3000</seg_maxVelikost>
    <seg_minVelikost>15</seg_minVelikost>
    <seg_minPocetNalezenych>3</seg_minPocetNalezenych>
    <seg_maxVzdalenostSlov>18</seg_maxVzdalenostSlov>
    <seg_musiObsahovat>Tdatum</seg_musiObsahovat>
    <seg_musiObsahovat>Tcena</seg_musiObsahovat>
    <seg_slovo>dnu</seg_slovo>
    <seg_slovo>snidane</seg_slovo>
    <seg_slovo>bez</seg_slovo>
```

[10] http://www.jakpsatweb.cz/html/rejstrik.html

```
        <seg_slovo>polopenze</seg_slovo>
        <seg_token>Tdatum</seg_token>
        <seg_token>Tcena</seg_token>
        <seg_token>Tteplota</seg_token>
        <seg_tag>tr</seg_tag>
        <seg_tag>a</seg_tag>
        <seg_tag>ul</seg_tag>
    </segment>
</vzor>
```

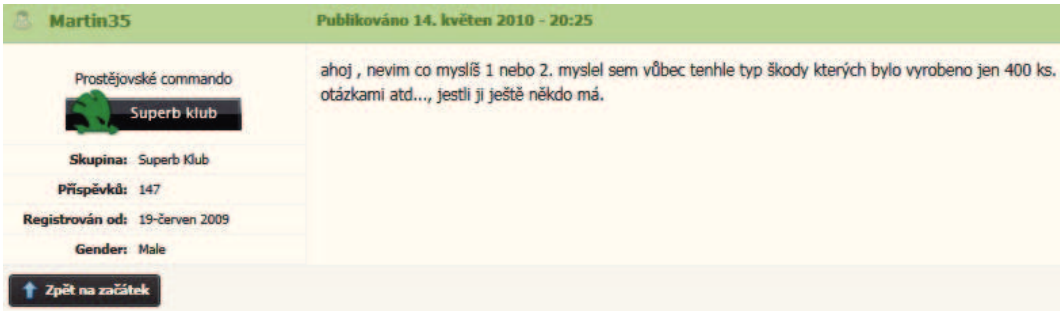
Výpis 1: Příklad XML

Popis parametrů XML: Parametry označené * jsou povinné parametry a jejich hodnota musí být nastavena na přípustnou hodnotu.

- `<jmeno>`* = název domény
- `<segment id = "nabidka">`* = název vzoru
- `<seg_hodnota>`* = váha vzoru
- `<seg_maxVelikost>`* = maximální velikost textu ve znacích
- `<seg_minVelikost>`* = minimální velikost textu ve znacích
- `<seg_minPocetNalezenych>`* = minimální počet klíčových slov nalezených v textu potřebných pro rozpoznání vzoru. Stejné slovo se započítá jenom jednou.
- `<seg_maxVzdalenostSlov>`* = vzdálenost od prvního nalezeného slova do posledního. Vzdálenost ve slovech. Pokud se jedná o vzor, který se v textu může vyskytnou vícekrát, musím hodnotu nastavit na kladné číslo (Viz kapitola 6.10.2). Pokud vyhledávám vzor, který se v textu vyskytuje jedenkrát, hodnotu nastavím na -1 (Viz kapitola 6.10.1).
- `<seg_musiObsahovat>` = slovo, které musí vzor obsahovat vždy. Tedy bez něho vzor nebude nalezen. Toto slovo musí být uvedeno i v `<seg_slovo>`.
- `<seg_slovo>`* = klíčová slova, která mohou, ale nemusejí být obsažena. Tato slova jsou jedním z nejdůležitějším parametrů. Jejich počet není omezen.
- `<seg_token>`* = datový typ který může nebo nemusí být obsažen. Datové typy velice pomáhají při analýze. Bez datových typů by většina vzorů nešla efektivně vyhledat. Počet není omezen.

- <seg_tag>* = tag HTML. Obvykle více TAGů. Identifikuje, kde se může vyskytovat vzor. Např. pokud zvolím tag <title>, budu vzor vyhledávat pouze v titulku stránky. Na rozdíl TAG <body> znamená, že vzor budu vyhledávat přes celou webovou stránku.

5.4 Obrázky vybraných návrhových vzorů



Obrázek 5: Návrhový vzor Diskuze

paměť	Paměť DDR3 3 GB (1 x 1024 + 1 x 2048 MB)
Maximální paměť	Podpora 8 GB paměti DDR3

Obrázek 6: Návrhový vzor Technické údaje

Obrázek 7: Návrhový vzor Počasí

6 Návrh ALGORITMŮ

Celý program funguje na následující myšlence. Na analyzované stránce aplikace vyhledává webové vzory, které jsou charakteristické pro doménu. Pokud takový vzor nalezl, zaznamená to a inkrementuje počet. Pro výpočet hodnoty podobnosti používám Kosinovou míru. Informace o vzorech ukládám do vektorového modelu. Funkce programu je popsána pomocí sady heuristik:

- Pomocí aplikace HTML Agility Pack (HAP) předveď stránku na XML.
- Postupně procházej seznamu TAGů, vyber jeden a pokud se na stránce vyskytuje, získej text, který obsahuje.
- Tento text vyčisti a zjisti zda má správnou velikost (odmazání konce řádku, tabulátory atd...)
- Zjisti zda se text poprvé analyzuje, pokud ne, přeskoč.
- Příprava textu - *Tokenizace* (určí hranice slov, oddělení textu od číslovek a od znamének čárka, tečka, vykřičník atd.).
- Větu rozděl na tokeny, převeď na malá písmena a odstraň diakritiku.
- Poznej datové typy první průchod. (typ: *číslo, malé číslo, datum, destinace, měsíc*).
- Odstraň koncovky.
- Poznej datové typy druhý průchod. (typ: cena, teplota, jednotky obecně typy, které se skládají z více tokenů).
- Analýza pro vzory, které se v TAGu vyskytují jednou (titulek).
- Analýza pro vzory, které se v TAGu vyskytují vícekrát (teplota, technické údaje).
- Výpočet podobnosti vektoru pomocí Kosinové míry

Jednotlivé kroky vysvětlím podrobně v následujících kapitolách. Uvedu příklady zdrojového kódu. Jednodušší příklady přímo v jazyku C#. Složitější v pseudokódu.

6.1 HTML Agility Pack (HAP)

Pro získání XML souboru jsem použil již vytvořenou aplikaci HAP [11]. HAP je agilní HTML parser, který zapisuje nebo čte DOM, a podporuje XPAth nebo XSLT. Je naprogramovaný v .NET knihovně, která umožňuje analyzovat z webu HTML soubory. HAP je velmi tolerantní k chybně napsaným HTML. Objektový model je velmi podobný, co navrhuje System.Xml, ale pro HTML dokumenty nebo streamy. Pomoci HAP převádím soubory HTM do XML. Tento XML soubor slouží jako vstup pro další krok algoritmu.

6.2 HTML TAG v XML souborech

Při načítání z konfiguračního XML u každého segmentu zaznamenej v jakém TAGu se může vyskytovat. Tyto údaje ulož do pole `seznamTagu` tímto způsobem: jestliže `Tag` není obsažen v poli `seznamTagu` přidej ho.

```
if (!vzor.seznamTagu.Contains(tag.InnerText))
  vzor.seznamTagu.Add(tag.InnerText);
```

Výpis 2: Seznam tagů

Pole `seznamTagu` procházej a jednotlivé TAGy vyhledej v XML. Výsledek ulož do `XmlNodeList`. Procházej `XmlNodeList` a zjisti zda obsahuje text, pokud ano pokračuj dalším krokem.

```
XmlDocument dok = new XmlDocument();
XmlElement root = dok.DocumentElement;
XmlNodeList shoda = root.GetElementsByTagName(tag);
```

Výpis 3: Seznam kořenů

Třída `XmlDocument` představuje dokument jako obrácený strom uzlů, kořenový element je na vrcholu. Každý uzel je instancí třídy XmlNode definující metody a vlastnosti pro procházení stromy DOM(Dokument Object Model).

6.3 Smazání a nahrazení v textu

Text získaný v předchozím kroku může obsahovat formátovací značky používané programátory html. Je potřeba je nahradit jiným znakem. Např. `\n` nahradím znakem `SPC`[12].

```
veta = veta.Replace('\n', ' ') ;
```

[11] http://htmlagilitypack.codeplex.com/
[12] SPC - space, mezera, "prázdný znak"

Text v HTML obsahuje entity. Entita je textová reprezentace určitého znaku. V HTML se používá mnoho roztodivných znaků, které však nelze buď přímo zapsat, nebo je jejich přímý zápis vyhrazen k jiným účelům. Entita se zapisuje ve tvaru `&jmeno_entity;`. Abych mohl entitu správně rozpoznat, musím ji nahradit příslušným znakem, pokud je to možné. Např. `&#186` nahradím znakem °.

```
s˜= s.Replace("&#186;", "°");
```

Jako poslední z textu mažu úseky textu, které nemají žádné, pro mě důležité informace. Pro tento úkol jsem vytvořil metodu `Odmaz(zdroj,zacatek,konec)`. Tato metoda zajistí, že ve zdroji odstraní všechen text začínající značkou `začátek` a končící značkou `konec`. Nejčastěji se v textu vyskytuje skript, který začíná značkou `//` a končí `////`.

```
private string odmaz(string s, string start, string konec)
{
    int delkaStart = start.Length;
    int delkakonec = konec.Length;
    int startPozice = 0;
    int konecPozice = 0;
    while (s.IndexOf(start, startPozice) != -1)
    {
        int index = s.IndexOf(start, startPozice);
        startPozice = index;
        index = s.IndexOf(konec, startPozice + delkaStart);
        if (index != -1)
        {
            konecPozice = index;
        }
        else break;
        s˜= s.Remove(startPozice, konecPozice - startPozice + delkakonec);
    }
    return s;
}
```

Výpis 4: Mazání v textu

6.4 Redundance v HTML

V HTML jazyku se stává, že text bývá zabalen do několika TAGů. Uvedu příklad:

```
<tbody>
    <tr>
        bydliste: Praha
```

```
        zalozen: 19.11.2000
    </tr>
</tbody>
```

Výpis 5: Redundance v HTML

Pokud budu vyhledávat TAG `<tbody>` obdržím text *Praha 19.11.2000*. Stejně tak při vyhledání `<tr>` obdržím text *Praha 19.11.2000*. Pokud by tento text vyhovoval všem požadavkům, započítal by se 2x a to není přípustné. Proto musí existovat pole `textZpracovany`, které procházím sekvenčně. A v něm si ověřím, zda text byl zpracovaný či nikoliv.

```
text.Contains(textZpracovany) || textZpracovany.Contains(text)
```

6.5 Rozdělení na tokeny

Základním stavebním kamenem pro analýzu jsou části textů, které se nazývají tokeny a velmi často odpovídají rozdělení na slova. Problém je najít hranice těchto pozic. Texty neobsahují jen písmena (malá, velká), ale i speciální znaky a číslice. V HTML jazyku slova nemusejí být odděleny mezerou. Uvedu příklad takové věty: *nákupuCeny 20-10-2008platby,ProdejnyVaše cena:2000,-Kč*. Tato věta by měla být rozdělena na tokeny: *nákupu Ceny 20-10-2008 platby , Prodejny Vaše cena : 2000,- Kč*. Slova se rozdělí na základě malých a velkých písmen, číslovek a interpunkčních znamének. Pro tento účel jsem naprogramoval metodu `pripravText(veta)`. V příloze uvedu část kódu, který se týká slov tvořených velkým písmem. Obdobný kód je pro malá písmena a číslovky. Kód může obsahovat vyjímky př. znaky , + - může znamenat cenu ,-, proto je nechám neoddělené mezerou. Dalším příkladem může být datum "20-10-2008", který zůstane taktéž nerozdělen. Zjednodušený zdrojový kód jsem uvedl v příloze A.1.

Ze zdrojového kódu (příloha A.1) je patrné, že cyklus prochází znak po znaku a kontroluje, jakého je znak typu. Po určení typu se testuje následující znak již na připravené podmínky. Jednotlivá slova se ukládají do proměnné `p`. Jakmile se zjistí konec slova, uloží se do výsledné proměnné `slova` a pokračuje se od začátku výběrem typu. Typy jsou velké, malé a číslice. Jako předposlední se testuje mezera, pokud předchozí znak nebyla mezera, vložím ji. Poslední otestuji, zda znak není mezera, písmeno nebo číslice. Pokud tomu tak je, vložím znak a za něj oddělující mezeru. Rozdělená věta je uložena do datového typu string.

6.6 Odstranění diakritiky, Stop slova

Jako vstup slouží věta z předchozího příkladu. Pro potřeby analýzy je vhodné převést písmena na malé a následně odstranit diakritiku. Toto odstranění se většinou řeší výčtem všech znaků a následným nahrazováním za adekvátní. Naštěstí existuje lepší řešení. Pro odstranění diakritiky jsem vytvořil metodu `OdstraneniDiakritiky(veta)`.

```
veta = veta.ToLower();
public static string OdstraneniDiakritiky(String veta)
{
//oddělení znaků od modifikátorů (háčků, čárek, atd.)
veta = veta.Normalize(System.Text.NormalizationForm.FormD);
System.Text.StringBuilder sb = new System.Text.StringBuilder();
    for (int i = 0; i < veta.Length; i++)
    {
        // do řetězce přidá všechny znaky kromě modifikátorů
        if (System.Globalization.CharUnicodeInfo.GetUnicodeCategory(veta[i]) != System.
            Globalization.UnicodeCategory.NonSpacingMark)
        {
            sb.Append(veta[i]);
        }
    }
    // vrátí řetězec bez diakritiky
    return sb.ToString();
}
```

Výpis 6: Odstranění diakritiky

Výpis 6 ukazuje jak je tato metoda velmi jednoduchá. Znaková sada unicode nabízí řadu funkcí, se kterými lze snadno diakritiku oddělit a zrušit. Nejdříve se normalizuje text do podoby `Normalize(System.Text.NormalizationForm.FormD)`. Znaky, které jsou označovány jako `NonSpacingMark`, se umístí do textu samostatně (Př. *žena* se převede na *zˇena*). Pak stačí větu projít a do výsledku nezařadit `NonSpacingMark`.

Tuto větu bez diakritiky zpracuji a rozdělím na slova. Text rozdělím podle mezer. Tímto jsem dostal pole jednotlivých slov. Při tomto rozdělení testuji, zda slovo není obsaženo v seznamu tzv. Stop list. Pokud provedu frekvenční analýzu jakékoliv webové stránky, zjistím, že nejfrekventovanějšími termy jsou spojky, předložky, interpunkční znaménka a další výrazy. Z těchto termů vytvořím Stop list. Pokud je testované slovo obsaženo ve Stop listu vynechám ho. Viz výpis 7.

```
List<string> list = new List<string>();
foreach (string slovo in upraveno.Split(' '))
{
    if (stopSlova.Contains(slovo.Trim()))
```

```
        continue;
        list.Add(slovo);
    }
```

Výpis 7: Rozdělení věty podle mezer

6.7 Datové typy - první průchod

V tomto kroku projdu všechna slova v poli a zjišťuji jejich datový typ. V prvním průchodu testuji základní typy jako jsou číslovky, měsíce a destinace. Měsíce a destinace jsou načteny ze souboru. Z těchto základních datových typů v druhém průchodu (Viz. kapitola 6.9) vytvořím složitější typy. Pokud slovo nevyhovuje žádnému testovacímu případu, uložím ho do výsledku jako obyčejný text. Pokud je slovo číslovka, spočítám počet číslic a počet oddělovačů Př. *10, 2008, 10-12-2008*. Na základě těchto údajů určím typ. Viz zjednodušený výpis 8.

```
int pocitadlo = 0;
int oddelovac = 0;
for (int t = 0; t < slovoDelka; t++)
{
    if (char.IsDigit(slovo[t]))
    {
        pocitadlo++; continue;
    }
    if (slovo[t] == '.' || slovo[t] == '-')
    {
        oddelovac++; continue;
    }
    else break;
}
```

Výpis 8: Počet číslic

Na základě počtu oddělovačů a cifer určím datový typ. U každého typu uvedu příklady a použití.

Datový typ `Ttelefon`. Telefonní číslo obsahuje 9 nebo 12 číslic. Uvažuji mobilní telefony. Př: *605600600, 420605600600*. Tento typ se dá využít pro hledání kontaktů. Pozn. Tento kód uvedu kompletní, další příklady již velice zjednodušené.

```
if ((pocitadlo == 9 || pocitadlo == 12) && oddelovac == 0)
{
    token.Add(new Token(Token.TYP.Ttelefon, slovo));
    pridano = true;
```

```
        continue;
    }
```

Výpis 9: Datový typ telefonní číslo

Datový typ `TmaleCislo`. Malé číslo obsahuje maximálně 2 číslice a oddělovač je maximálně 1. Př: *12, 1.2*. Tento typ využiji pro hledání pořadových čísel nebo při hledání měsíce (Př. 12 srpna).

```
if (pocitadlo <= 2 && pocitadlo <= 1)
```

Datový typ `Tcislo`. Velké číslo obsahuje více než 3 číslice a oddělovač je maximálně 1. Př: 25000, 3 000.00. Tento typ využiji při hledání ceny.

```
if (pocitadlo >= 3 && oddelovac <= 1)
```

Datový typ `Tdatum`. Datum obsahuje více než 2 číslice a oddělovač je v rozmezí 2 až 5 včetně. Př: 2.10., 25.1.-25.4.2005. Typ datum využiji při hledání vzoru dovolená nebo diskuse.

```
if (pocitadlo >= 3 && oddelovac >= 2 && oddelovac <= 5)
```

Datový typ `Tdestinace`. V následujícím případu testuji slovo na to, jestli je to destinace. Destinací rozumím místo dovolené, může to být stát, ostrov nebo letovisko. Tyto destinace načítám ze souboru. V souboru jsou destinace uloženy bez diakritiky. Abych našel co nejvíc výskytů, odstraním ze slov koncovky - tento algoritmus popíšu v kapitole 6.8. Pokud se slovo bez koncovky shoduje s nějakým slovem bez koncovky ze seznamu, označím ho jako destinaci. Př: Omán, Německo, Vanuatu. Tento typ využiji při hledání vzoru dovolená.

```
if ( statyList .Contains(slovoBezKoncovky))
```

Datový typ `Tmesic`. Poslední testovací případ hledá měsíce, které jsou obdobně jako předchozí případ uloženy v souboru bez diakritiky. Př: leden, led, ledna. Tento typ využiji při hledání datumu.

```
if (mesiceList.Contains(slovo))
```

6.8 Odstranění koncovky - kmen slova

Čeština má mnoho tvarů slov. Abych tento problém alespoň z části eliminoval, vytvořil jsem množinu koncovek, které odtrhávám od slov a tím získám kmen slova. Koncovka je zakončení ohebných slov, které se při skloňování nebo časování slova mění. Na rozdíl

od předpony nebo přípony nevytváří nové slovo, ale vytváří různé tvary téhož slova. V některých tvarech může koncovka chybět (tzv. nulová koncovka). Vytvořil jsem algoritmus, který odtrhává koncovky od slov. Tyto koncovky, které odtrhávám, jsou uloženy v souboru. Algoritmus je jednoduchý. Funguje tak, že procházím seznam koncovek a každou z nich se pokouším odtrhnout. Pokud nějakou odtrhnu, vracím kmen. Př: *kovovy - ovy = kov*.

```
public string odstranKoncovku(string s)
{
    string orezane = String.Empty;
    bool nasel = false;
    foreach (string koncovka in koncovkyList)
    {
        if (s.EndsWith(koncovka))
        {
            int delkaSlova = s.Length;
            int delkaKoncovky = koncovka.Length;
            orezane = s.Substring(0, delkaSlova − delkaKoncovky);
            nasel = true;
            break;
        }
        else nasel = false;
    }
    if (nasel) return orezane;
    else return s;
}
```

Výpis 10: Odstranění koncovky

6.9 Datové typy - druhý průchod

Po prvním průchodu, jak byl uveden v kapitole 6.7, následuje průchod druhý. Tento průchod přes všechny tokeny vyhledává v textu případy, které obvykle spojují dva tokeny do jednoho a určí jeho typ. Uvedu malý příklad. V kapitole 6.5 jsem věty rozdělil na slova a značky. Pokud potřebuji vyhledávat slovo *od:* je zapotřebí spojit token *od* + token *:*. Výsledkem je `token(Ttext,od:)`. Na tomto příkladu je patrné, že si musím pamatovat tokeny na několika pozicích. Jsou to `predchozi[i−1]`, `aktualni[i]`, `dalsi[i+1]`, `zaDalsim[i+2]`. Vytvořený token uložím do výsledku. Postupně popíšu všechny testovací případy.

Datový typ `Tcena`. Tento případ spojuje `Tcislo` + *Kc* nebo `Tcislo` + ,- + *Kc* a výsledek uloží jako `Tcena`. Viz následující zjednodušený kód. Př: 28000,-Kč, 10000,-. Tento datový typ využiji při identifikaci vzoru nabídka dovolené.

```
if (token[i].nazev == Token.TYP.Tcislo)
{
    if (dalsi.text.Contains("kc") || dalsi.text.Contains("czk"))
    {
        Token t = new Token(Token.TYP.Tcena, textPredchozi + token[i].text + dalsi.text);
    }
    if (dalsi.text == ",-")
    {
        if (zaDalsim.text.Contains("kc") || zaDalsim.text.Contains("czk"))
        {
        Token t = new Token(Token.TYP.Tcena, textPredchozi + token[i].text + dalsi.text + zaDalsim.
            text);
        }
        else
        {
            Token t = new Token(Token.TYP.Tcena, textPredchozi + token[i].text + dalsi.text);
        }
    }
}
```

Výpis 11: Token cena

Datový typ `Tteplota`. Následující kód poznává datový typ teplota. Teplota se skládá z `TmaleCislo` + ° + *Celsia*. Př: 25°C, 15°c. Tento datový typ využiji při hledání vzoru teplota.

```
if (token[i].nazev == Token.TYP.TmaleCislo)
{
    if (dalsi.text == "°")
    {
        if (zaDalsim.text == "C" || zaDalsim.text == "c")
        {
            Token t = new Token(Token.TYP.Tteplota, token[i].text + dalsi.text + zaDalsim.text);
        }
        else
        {
            Token t = new Token(Token.TYP.Tteplota, token[i].text + dalsi.text);
        }
    }
}
```

Výpis 12: Token teplota

Datový typ `Tdatum`. V tomto testovacím případu hledám datum, který je tvořen malým číslem a měsícem, který je zapsaný slovem. Př: 17 lis, 20 ledna.

```
if (token[i].nazev == Token.TYP.TmaleCislo && dalsi.nazev == Token.TYP.Tmesic)
```

Datový typ `Tjednotka`. Tento testovací případ uvedu jen zjednodušeně. Jedná se o jednotku teploty. Již jsem uvedl `Tteplotu`, ale ta obsahovala i číslo. Tento příklad obsahuje jen ° a *Celsia*. Př °C, °c. Tento typ využiji pro vzor technické údaje.

```
if (((token[i].text == "°")) && (token[i + 1].text == "c") || (token[i + 1].text == "C"))
```

Datový typ `Tjednotka`. Následující kód testuje, zda text obsažený v tokenu je jednotka. Seznam jednotek je uložen v souboru. Př: km, mm, dpi, palce.

```
if (jednotkyList.Contains(token[i].text))
```

Abych mohl rozpoznat rozměry, vytvořil jsem testovací případ, který pozná rozměry typu *1024x768*. Je vhodné spojovat čísla oddělená / jako například *12/15/17*. Datový typ u spojených čísel zůstává stejný.

Tímto jsem se dostal k poslednímu problému a tím jsou jednotky, které jsou kombinací malých a velkých písmen. Je to například jednotka frekvence *KHz* nebo *dB*. Problém to je, protože v kapitole 6.5 rozděluji věty podle velikosti písmen. Slovo *KHz* se rozdělí jako *kh* a `z`. Slovo `z` neznamená jednotku, a tak musím vytvořit vyjímku, která pokrývá tento zvláštní případ. Př: Mhz

```
if (((token[i].text == "h") || (token[i].text == "kh") || (token[i].text == "mh") || (token[i
    ].text == "gh")) && ((token[i + 1].text == "z")))
```

6.10 Analýza

Nyní se dostávám k hlavnímu algoritmu celého programu. Je to vyhledávání klíčových slov z připraveného textu. Abych mohl aplikovat myšlenku, musím prohledávat text. Zde se dostávám k problému, že tento text může obsahovat jeden nebo více vzorů. Proto je nutné vytvořit dva algoritmy.

6.10.1 Vzor vyskytující se jednou

První z nich je algoritmus pro vyhledání několika klíčových slov z bloku textu. Tento počet slov je pro každý vzor jiný. Ve výpisu 13 hledám slovo, které se shoduje s některým z klíčových. Pokud takové najdu, zařadím ho do výsledku. Platí, pokud je počet

nalezených slov větší nebo rovno minimálnímu počtu, je vzor započítán. Je možné definovat žádné, nebo několik klíčových slov, které se musejí v daném bloku vyskytnout. Pokud je i tato podmínka splněna, pak tento blok označím jako vzor a inkrementuji jeho počet v datové struktuře vzor. Algoritmus má za úkol najít jeden vzor. Algoritmus porovnává kmeny slov a tím zvyšuje svoji účinnost. Demonstruji na pseudokódu:

```
1   List<string> Analyza(slova, pocitadlo, musiObsahovat)
2     DEF List<string> vysledek #zde ukladam nalezena klicova slova
3     DEF List<string> klicovaSlova #zde uložím klíčová slova
4     FOREACH(string slovo IN vzor[pocitadlo].seznamSlov)
5       slovo = odstranKoncovku(slovo)
6       VLOZ slovo DO klicovaSlova
7     FOREACH(Token token IN slova)
8       string s~= token.text
9       IF(klicovaSlova.Obsahuje(s) && !vysledek.Obsahuje(s)) THEN
10        VLOZ s~DO vysledek
11    DEF int min = vzor[pocitadlo].MinPocetNalezenych
12    IF(vysledek.Pocet >= min && obsahujePovinne(vysledek, musiObsahovat)) THEN
13      vzor[pocitadlo].Pocet += 1
14    ELSE THEN
15      CLEAR vysledek
16    RETURN vysledek
```

Výpis 13: Analýza

Ve výpisu 14 testuji, zda nalezená klíčová slova obsahují povinné KS. Postupně procházím seznam a pokud slovo naleznu, smažu ho. Jestliže je seznam prázdný, vracím true, jinak false.

```
1   bool obsahujePovinne(pole, slova)
2     DEF bool vysledek = false
3     FOREACH(string povinny IN slova)
4       IF(pole.Obsahuje(povinny)) THEN
5         ODEBER povinny ZE slova
6         continue
7     IF(slova.Pocitadlo == 0) THEN
8       vysledek = true
9     return vysledek
```

Výpis 14: Obsahuje povinné

6.10.2 Vzor vyskytující se vícekrát

Druhý z vyhledávacích algoritmů je založen na následující myšlence. V textu se vyskytuje více vzorů stejného druhu. Jako příklad poslouží vzor technické údaje. Tento vzor se

obvykle vyskytuje v textu (např. odstavec) více než jednou. Už mi nestačí najít ho jednou a započítat (viz. předchozí kapitola 6.10.1), ale musím najít a spočítat každý výskyt vzoru. Abych toho byl schopen, musím si určit maximální vzdálenost slov od sebe. Tuto vzdálenost definuji jako pozice posledně nalezeného - pozice prvního nalezeného <= maximální vzdálenost. Aby bylo vyhledání co nejpřesnější, musím vzor obsahovat povinná slova (nebo datové typy). U vzoru technické údaje to jsou povinná klíčové slova `Tjednotka` a `Tcislo`. Algoritmus je schopen vyhledat následující příklady: délka 50m, rozlišení 1024x768 pixelů, obsah 2000 cm3. Z důvodu rozsáhlého kódu uvedu zjednodušený pseudokód v příloze A.2. Výstup tohoto algoritmu je seznam, který obsahuje strukturu. V ní je zaznamenána věta a klíčová slova. V posledním kroku algoritmu přičtu počet záznamů do struktury vzor.

6.11 Podobnost vektorů

V kapitole 6.10 jsem zaznamenal frekvenci vzorů. Tím jsem získal jednoduchý systém vážení dokumentu (metoda TF). Dokument je popsán ve vektorovém modelu. Takový dokument si mohu představit jako n-tici. Dokument je vektor. Pro indexaci bylo použito celkem n různých vzorů.

$$D(V_1, V_2, \ldots, V_n), kde\ V_n \in N$$

V_n je frekvence vzoru, který se na stránce vyskytuje.

Výraz dotazu Q ve vektorovém modelu je možné formulovat jako n-místný vektor vah.

$$Q(Q_1, Q_2, \ldots, Q_n), kde\ Q_n \in R$$

Q_n je váha příslušného návrhového vzoru. Váha byla určena na základě pozorování, kolikrát se návrhový vzor vyskytuje na webových stránkách. Na základě dotazu lze spočítat podobnost s dokumentem. Výsledkem je koeficient podobnosti. Tento koeficient si lze představit jako podobnost vektoru dokumentu s vektorem dotazu. Pro výpočet podobnosti dokumentu D vzhledem k dotazu Q jsem použil Kosinovou míru. Viz. kapitola 3.2. Aby byl výsledek zobrazován v procentech musím výsledek vynásobit 100.

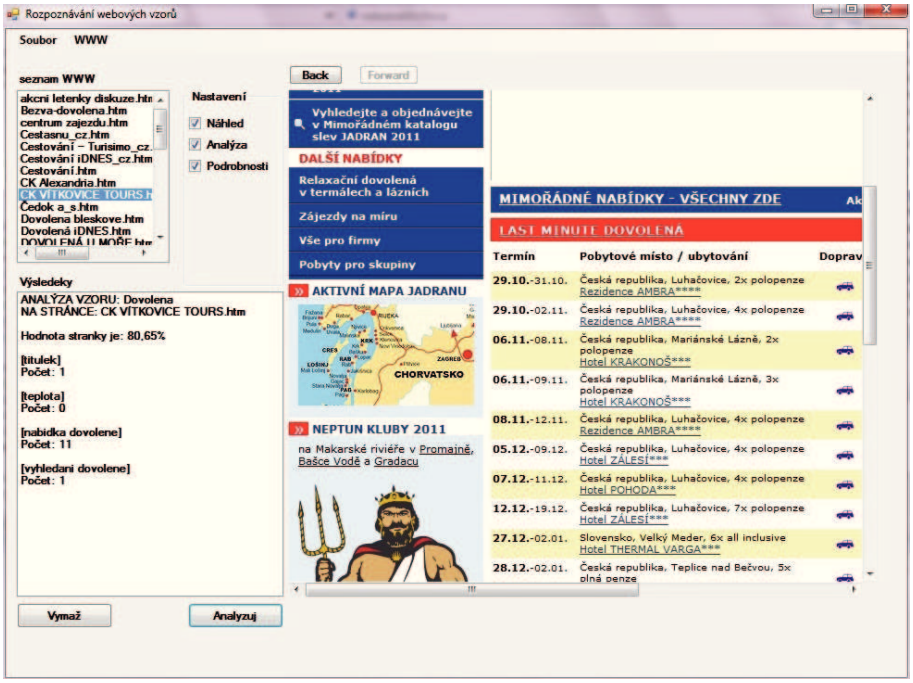
7 Návrh aplikace

Pro implementaci jsem si vybral programovací jazyk C#. Jako vývojové prostředí jsem použil Microsoft Visual Studio 2008. Verzi jazyka C# jsem použil Microsoft .NET framework 3.5.

Program běží v grafickém okně. Popis jednotlivých prvků je uveden v uživatelské příručce, která je součástí přílohy. V uživatelské příručce je dále uveden postup spuštění programu a minimální konfigurace. Programátorská příručka obsahuje popis všech tříd a metod (datové typy, návratové hodnoty atd.). Příručka je součásti přiloženého CD. Součástí přiloženého CD je i Třídní Diagram a diagram sekvenční, který popisuje vyhledání informací v html dokumentu.

7.1 GUI

Program se spouští souborem `RWV.exe`. Jako první se načte grafické uživatelské rozhraní aplikace. To zajišťuje třída `GUI.cs`, která inicializuje komponenty GUI (prvky grafického rozhraní jsou ze třídy `System.Windows.Form`). Viz. obrázek 8. V následujícím textu uvedu všechny komponenty, které používám.



Obrázek 8: Uživatelské rozhraní programu RWV

Komponenta `ToolStripMenuItem` slouží pro zobrazení menu a jeho položek. Pro popis komponent používám `Label`. V aplikaci je potřeba zobrazovat načtené dokumenty a výsledky. K tomuto účelu používám `ListBox`. Pro nastavení funkčnosti programu jsem použil komponentu `CheckBox` (funkce náhled, podrobnosti a analýza). Aplikace má za úkol analyzovat html stránky. Pro zobrazení těchto stránek jsem implementoval www prohlížeč. V jazyku C# použiji komponentu `WebBrowser`. Poslední komponentou je obyčejné tlačítko `Button`, které využívám ke spuštění analýzy, vymazání výsledků a k navigaci.

7.2 Načtení vstupních dat

Pro očekávanou funkci programu je nutné načíst vstupní data. Jedná se o načtení vzoru, který chci vyhledat, a načtení www stránek, které budu analyzovat. Pro načtení vstupních dat používám komponentu `OpenFileDialog`. V OpenFileDialogu jsem nastavil složku, ve které se dialog otevře. Dále nastavím typy souborů, které budu otevírat. Pro načtení vzoru to jsou soubory s koncovkou .xml a pro načtení www stránek to jsou soubory s koncovkou .html, .htm, .mht. O načtení vzoru stará funkce `nactiZeSouboru2(string cesta)`. Tato funkce projde všechny záznamy v XML souboru a uloží je do připravené třídy `Vzor`. Třída `vzor` mimo jiné obsahuje seznam, do kterého se vloží jednotlivé návrhové vzory (nazývám je segmenty třída `Segment`), které popisují doménu. Tyto segmenty obsahují mnoho informací např. seznam klíčových slov, seznam tagů, seznam datových typu atd. Kompletní diagram tříd je uveden na přiloženém CD. O načtení stránek se stará dialog, který po úspěšném načtení zobrazí názvy www stránek do „seznamuWWW“ (`ListBox`). Tímto krokem aplikace načetla vše potřebné a nyní může vyhledávat informace.

7.3 Vyhledání informací

Před vyhledáním je nutné označit jeden html dokument. O to se stará funkce `seznamWWW_SelectedIndexChanged()`, která nastaví cestu, a pokud je zakliknutý checkBox „náhled“, zobrazí stránku ve webovém prohlížeči. Program nyní provede frekvenční analýzu nebo vyhledání informací. Vyhledání informací provede, když je zakliknutý checkBox „Analýza“, v opačném případě provede frekvenční analýzu.

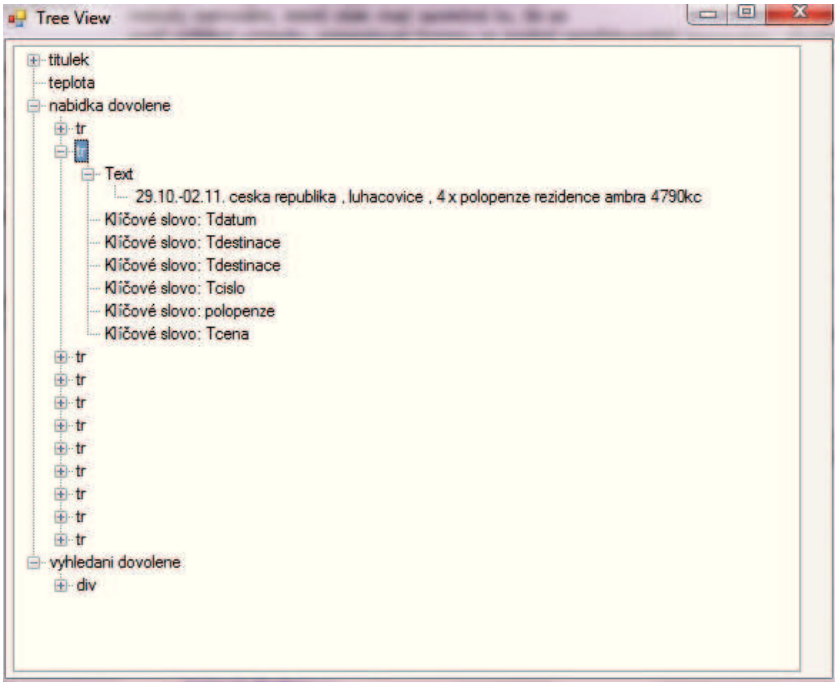
Frekvenční analýza: před provedením frekvenční analýzy je nutné vytvořit textový soubor, který obsahuje pouze čistý text z html dokumentu. O to se stará třída `HtmlToText`, která je součástí HAP. Funkce `Convert()` vytvoří soubor, který obsahuje pouze text. Tento text předám třídě `frekvencniAnalyza`,

která pomocí metody `vypocitejCetnosi()` provede rozdělení textu na slova (`Regex.Matches(upraveno, @"\w+")`). Pomocí funkce `ToLower()` převedu text na malá písmena a odstraním koncovky `odstranKoncovku()` a diakritiku. Jednotlivá slova ukládám do slovníku `Dictionary<string, int>()`. Pokud se slovo zopakuje, inkrementuji počet. Výsledný slovník zobrazím (`ListBox` Výsledek) pomocí funkce `vypisNaObrazovku()`. Výsledek je setříděný. Výsledná slova jsou zobrazena bez koncovek.

Vyhledání návrhového vzoru: inicializačním krokem je vytvoření XML souboru. O to se stará HAP. HTML soubor načtu do třídy `HtmlDocument` nastavím parametr `OptionOutputAsXml` na `true` a soubor uložím pomocí funkce `Save()`. Tento dokument předám třídě `TagXml`, která pomocí funkce `Analyzuj()` provede analýzu textu. Tato funkce zajistí:

- načtení XML dokumentu do třídy `XmlDocument`
- výběr tagů, které se budou prohledávat `GetElementsByTagName()`
- odstranění nepotřebných částí textu `odmaz()`
- přípravou textu:
  1. funkce `pripravText()` upraví text do takové podoby, aby byl od sebe oddělen obyčejný text, znaménka a číslice. Tato slova jsou od sebe oddělena mezerou.
  2. o odstranění diakritiky se stará funkce `OdstraneniDiakritiky()`, text je rozdělen na slova a je převeden na malá písmena. Slova jsou uložena do `List<string>`
  3. funkce `poznejTyp()` prochází slova a snaží se rozpoznat datové typy. Výčet datových typů obsahuje třída `Token`.
  4. funkce `poznejCenu()` rovněž prochází slova s tím rozdílem, že se snaží naleznout složitější datové typy. Výčet datových typů obsahuje třída `Token`.
- připravený text předám třídě `Analyza`, kde funkce `cetnosti()` zjistí, kolikrát je v bloku textu obsažen návrhový vzor. Třída obsahuje funkci `vypocitejCetnosti()`, která vyhledává pouze jednu instanci návrhového vzoru. Výběr funkce je nastaven v souboru XML. Výsledkem je seznam `List<strukturaPV>`, který obsahuje nalezená klíčová slova, datové typy a blok textu, ve kterém byl nalezen.

- zobrazení výsledků obstarává funkce `vypisNaObrazovku()`. Program dokáže zobrazit podrobnosti vyhledávání. V průběhu vyhledávání si vytvářím stromovou strukturu nalezených segmentů - třída `Tag`. Po ukončení procesu vyhledávání se otevře okno (`treeWindow`), ve kterém je zobrazen seznam segmentů(návrhových vzorů). Každý segment obsahuje seznam tagů, ve kterých byl nalezen. Každý tag obsahuje seznam nalezených klíčových slov a text. Viz. obrázek 9. Proces vyhledání návrhového vzoru jsem zachytil v sekvenčním diagramu, který je umístěn na přiloženém CD ve složce přílohy. Kompletní diagram tříd je také umístěn na přiložením CD ve složce přílohy.



Obrázek 9: Okno s podrobnostmi

8 Experimenty a vyhodnocení výsledků

Pro hodnocení úspěšnosti programu jsem použil testovací data, která jsem použil pro analýzu, a podle kterých jsem nastavoval parametry programu. Tato data jsem rozdělil do třech skupin a to podle domény (Dovolená, Diskuse, Technické parametry). Každá skupina obsahuje stránky pozitivní i negativní. V každé doméně vyhledávám určité vzory. Viz následující seznam.

1. **Dovolená**: titulek, teplota, nabídka dovolené, vyhledání dovolené
2. **Diskuse**: titulek, příspěvek
3. **Technické údaje**: titulek, technický popis

Abych mohl určit úspěšnost, musím si pro každý vzor vytvořit tzv. matici záměn. V tabulce 5 je zaznamenáno v kolika případech se shoduje ruční vyhledání s automatickým vyhledáním. Ve sloupci jsou informace jak vyhledával program, v řádcích je informace o ručním vyhledání.

	automaticky	
ručně	**+**	**-**
+	TP	FN
-	FP	TN

Tabulka 5: Matice záměn

Matice změn pouze zaznamenává dobře a špatně zařazené příklady. Vysvětlení hodnot v tabulce.

1. **TP (true positive)** - počet příkladů, které jsou správně automaticky zařazeny do positive(+).
2. **FP (false positive)** - počet příkladů, které jsou chybně automaticky zařazeny do třídy (+).
3. **TN (true negative)** - počet příkladů, které jsou správně automaticky zařazeny do třídy (-).
4. **FN (false negative)** - počet příkladů, které jsou chybně automaticky zařazeny do třídy (-).

Použité vzorce:

- **Pozitivní přesnost =** $\frac{TP}{TP+FP}$. Popisuje kolik pozitivních nalezených dokumentů se skutečně týká tématu (např. tématu dovolená).
- **Negativní přesnost =** $\frac{TN}{TN+FN}$. Popisuje kolik negativních nalezených dokumentů se netýká tématu.
- **Úplnost =** $\frac{TP}{TP+FN}$. Popisuje kolik dokumentů týkajících se tématu jsme nalezli.
- **F-míra =** $\frac{2*pozitivni\ presnost * uplnost}{pozitivni\ presnost + uplnost}$. F-míra je souhrnná charakteristika. Kombinuje pozitivní přesnost a úplnost. F-míra je nejčastěji používaná metrika pro hodnocení klasifikátorů.

Data, která jsem použil pro vytvoření vzorů, jsem použil i pro testování úspěšnosti, proto musím dobře popsat chybovost aplikace. Jak popsat chyby FP a FN? Zvolil jsem následující postup:

1. Pokud počet ručně ohodnocených >0 a zároveň je počet automaticky ohodnocených stejný, potom inkrementuji proměnnou TP.
2. Pokud počet ručně ohodnocených >0 a zároveň je počet automaticky ohodnocených menší, potom inkrementuji proměnnou FN.
3. Pokud počet ručně ohodnocených >0 a zároveň je počet automaticky ohodnocených větší, potom inkrementuji proměnnou FP.
4. Pokud počet ručně ohodnocených $=0$ a zároveň je počet automaticky ohodnocených stejný, potom inkrementuji proměnnou TN.
5. Pokud počet ručně ohodnocených $=0$ a zároveň je počet automaticky ohodnocených větší, potom inkrementuji proměnnou FP.

8.1 Výsledky pro doménu Dovolená

V doméně dovolená se vyskytují 4 vzory: *Titulek, Nabídka dovolené, Vyhledání dovolené Teplota v destinacích*. Pro každý uvedu matici záměn.

V tabulce 6 je matice záměn pro *Titulek*. Z tabulky je patrné, že titulek jsem našel ve všech případech a ve všech byl počet zcela správný. Titulek se vyskytuje jen ve dvou možnostech (ANO/NE), což je podle mého očekávání (titulek se na stránce vyskytuje pouze jednou nebo vůbec). Viz. tabulka

V tabulce 7 je matice záměn pro *Nabídka dovolené*. Z tabulky je patrné, že ne na všech stránkách aplikace našla správný počet. To je obvykle způsobeno nesprávným pojmenováním (překlep) nebo špatným naformátováním. Na tomto místě by bylo dobré

	automaticky	
ručně	**+**	**-**
+	26	0
-	0	8

Tabulka 6: Matice záměn: titulek

uvést, že vzor *Nabídka dovolené* musí obsahovat cenu zájezdu a datum zájezdu. Pokud to není splněno vzor se nezapočítá (ručně i automaticky). Zmiňuji to proto, že existují stránky, které obsahují podobný prvek jako nabídka dovolené, ale obvykle neobsahují datum nebo i cenu (tento prvek není vzor Nabídka dovolené, ale jakoby reklama).

	automaticky	
ručně	**+**	**-**
+	17	3
-	1	13

Tabulka 7: Matice záměn: Nabídka dovolené

V tabulce 8 je matice záměn pro *Vyhledání dovolené*. Z tabulky je patrné, že aplikace občas nalezne o jeden výskyt více než ručně. Je to způsobeno tím, že vzor *Nabídka dobolené* a *Vyhledání dovolené* mají mnoho společných termů. Dá se říci, že vyhledání a nabídka spolu souvisí. Parametry, které zadáme do vyhledání se objeví v nabídce. U vzoru nabídka jsem použil hlavně datové typy a výčet možností (např. datový typ datum a cena, letecky, busem, polopenze, snídaně, večeře atd). U vzoru vyhledání jsem použil názvy proměnných (doprava, strava, datum atd.) Toto je důvod chybovosti, která je patrná z tabulky 8.

	automaticky	
ručně	**+**	**-**
+	18	0
-	5	11

Tabulka 8: Matice záměn: Vyhledání dovolené

V tabulce 9 je matice záměn pro vzor *Teplota*. Tento vzor má 100% úspěšnost nalezení. Pokud se teplota na stránce vyskytovala byla nalezena zcela přesně.

	automaticky	
ručně	**+**	**-**
+	6	0
-	0	28

Tabulka 9: Matice záměn: Teplota

V tabulce 10 uvedu souhrnné charakteristiky pro doménu Dovolená. Hodnoty jsem objasnil v předcházejícím textu.

	Titulek	**Nabídka**	**Vyhledání**	**Teplota**
P. přesnost	1	0,944	0,783	1
N. přesnost	1	0,813	1	1
Úplnost	1	0,85	1	1
F-míra	1	0,895	0,878	1

Tabulka 10: Dovolená souhrnné výsledky

8.2 Výsledky pro doménu Diskuse

V doméně Diskuse vyhledávám dva vzory: *Titulek a Příspěvek do diskuse* . V tabulce 11 je matice záměn pro *Titulek*. Titulek byl 100% nalezen.

	automaticky	
ručně	**+**	**-**
+	16	0
-	0	13

Tabulka 11: Matice záměn: titulek

V tabulce 12 je uvedena matice záměn pro vzor *Příspěvek do diskuse*. Z tabulky je patrná chybovost. Ta je způsobena především tím, že je někdy obtížné od sebe odlišit jednotlivé příspěvky. Záleží na druhu stránky a stylu, podle kterého je naprogramována. Rozdílnost je obvykle +- 1 příspěvek, což není vůbec špatné. Například, na některých stránkách se příspěvek vyskytuje 30 - 50 krát. Chybovost by byla menší, pokud by bylo možné rozpoznat jednotlivé přezdívky v diskusních fórech. To ale není možné, protože

přezdívky (nick) se nevytvářejí podle určitého pravidla (např. oorrbb, deacon, midwej). Občas příspěvek obsahuje jen velice málo informací (datum, přezdívka a text) z těchto údajů nejsem schopen poznat příspěvek.

	automaticky	
ručně	**+**	**-**
+	15	5
-	4	7

Tabulka 12: Matice záměn: Příspěvek

V tabulce 13 uvedu souhrnné charakteristiky pro doménu Diskuse. Z tabulky je patrné, že jako nejhůře vyšla negativní přesnost u příspěvku. To je způsobeno zvolenou metodikou. Pokud bych zvolil prahovou hodnotu, od které bych určil zda vzor je obsažen nebo není (např. pokud je počet větší než 10 stránka obsahuje diskusi) byl by výsledek mnohem lepší. Na druhou stranu by výsledky byly až moc dobré a nepopisovalo by to tak přesně chybovost.

	Titulek	**Příspěvek**
P. přesnost	1	0,789
N. přesnost	1	0,583
Úplnost	1	0,75
F-míra	1	0,769

Tabulka 13: Diskuse souhrnné výsledky

8.3 Výsledky pro doménu Technické údaje

Doména Technické údaje obsahuje dva vzory: *Titulek a Technický údaj*. V tabulce 14 je uvedena matice záměn pro vzor titulek. Vzor titulek jsem opět vyhledal ve všech případech správně.

V tabulce 15 je uvedena matice záměn pro vzor Technický údaj. Postupuji stejně jako v předchozích kapitolách s tím rozdílem, že uvažuji toleranci mezi ručním a automatickým ohodnocením. Toleranci jsem stanovil na 3 výskyty. Pokud se rozdíl mezi ručním a automatickým vyhledáním liší max o 3 výskyty beru to jako správný výsledek (inkrementuji TP nebo TN). Tuto toleranci jsem musel zavést, protože vzor *Technické údaje* je velice rozmanitý. Například: (váha 6 kg, šířka 6cm, rozlišení 2000 pixelů, frekvence

	automaticky	
ručně	**+**	**-**
+	23	0
-	0	5

Tabulka 14: Matice záměn: Titulek

2000GHz, 5km délka). Fyzikálních veličin je mnohem více než jsem uvedl a není možné je najít vždy všechny. Některé řádky se překrývají nebo se v jednom řádku najde více údajů. Často se tento vzor nevyskytuje v úplném tvaru (např. chybně popsaná veličina, údaje v řádku bez řádného popisu). Vzor technický údaj pro mě znamená trojici: *název jednotky + číslo + jednotka*, tato klíčová slova jsou u sebe. Z tabulky je patrné, že aplikace nedokáže vyhledat všechny technické údaje i když tam jsou. Tuto chybu bych opravil tím, že bych do definice návrhového vzoru přidal větší počet názvů jednotek. Dále bych rozšířil soubor, který obsahuje jednotky (např. m, cm, m2) o ještě více druhů a variant. Předpokládám, že aplikace by pak vyhledávala o něco lépe. Jsem si vědom, že to nikdy nebude dokonalé.

	automaticky	
ručně	**+**	**-**
+	14	10
-	0	4

Tabulka 15: Matice záměn: Technický údaj

V tabulce 16 jsem uvedl souhrnné charakteristiky pro doménu Technické údaje. Údaje v tabulce odpovídají chybovosti popsané v předchozím textu.

	Titulek	**Technický údaj**
P. přesnost	1	1
N. přesnost	1	0,28
Úplnost	1	0,583
F-míra	1	0,737

Tabulka 16: Technické údaje souhrnné výsledky

8.4 Hodnocení klasifikátoru

Pro hodnocení klasifikátoru jsem se rozhodl použít stejnou metodiku jako v předchozím textu (matice záměn). Data budou stejná jako v předchozích kapitolách 8.2, 8.1 a 8.3. Postup provádění byl následující:

- Vybral jsem doménu kterou budu testovat. Načtení dat, načtení vzoru.
- Pro každý soubor proveď:
- Ruční ohodnocení subjektivním způsobem (ano - stránka patří do domény dovolená, ne - nepatří)
- Automatické ohodnocení. Stanovím prahovou hodnotu, od které se rozhodnu, zda vzor byl nalezen respektive nebyl(prahová hodnota je 60%).
- Porovnání výsledku, vytvoření matic záměn.

V následujících třech kapitolách uvedu výsledky měření. V přehledných tabulkách vyhodnotím klasifikaci.

8.4.1 Klasifikace Dovolená

V tabulce 17 jsem uvedl matici záměn pro doménu Dovolená. Klasifikace dopadla zdařile. Největší chybu tvoří dokumenty, které sice patří do domény dovolená, ale neobsahují potřebné prvky pro kategorizaci. Pokud byla stránka správně nalezena (patří do skupiny TP), tak průměrně s přesností 80%.

	automaticky	
ručně	**+**	**-**
+	21	4
-	1	8

Tabulka 17: Matice záměn: klasifikace Dovolená

8.4.2 Klasifikace Diskuse

V tabulce 19 jsem uvedl matici záměn pro doménu Diskuse. Doménu diskuse vyhledávám s drobnou chybou. Důvodem takto malé chyby může být fakt, že vyhledávám menší počet vzorů (prakticky jen příspěvek do diskuse). Například zařazením vzoru

Rychlá odpověď (vzor se občas vyskytuje pod diskusí a slouží pro rychlé vložení odpovědi) bych rozšířil množinu hledaných vzorů, a tím bych mohl více upřesnit vektor dotazu pro doménu Dovolená. Jiné zlepšení by spočívalo ve stanovení minimálního počtu nalezených příspěvků (například 4), a tím bych zajistil dostatečný informativní charakter diskuse. Potom diskuse obsahující méně než 4 příspěvky by byla vyhodnocena jako negativní.

	automaticky	
ručně	**+**	**-**
+	22	0
-	2	7

Tabulka 18: Matice záměn: klasifikace Diskuse

8.4.3 Klasifikace Technické údaje

V tabulce 19 jsem uvedl matici záměn pro doménu Technické údaje. Chybovost je způsobena faktem, že klasifikátor určí za stránku za správnou, i když obsahuje jen jediný technický údaj. Řešením je již zmíněná prahová hodnota, pod kterou bude vzor vyhodnocen jako nenalezen.

	automaticky	
ručně	**+**	**-**
+	21	0
-	3	4

Tabulka 19: Matice záměn: klasifikace Diskuse

8.4.4 Souhrnné výsledky klasifikace

V tabulce 20 jsem uvedl všechny výsledky pohromadě. Výsledkem je přehledná tabulka, ve které je znázorněna úspěšnost klasifikátoru. Nejlépe klasifikátor rozpoznal diskusi a nejhůře dovolenou. Celkově klasifikaci hodnotím kladně. Aplikace dokázala zařadit většinu stránek do příslušné domény. Výsledky jsou velice dobré, tuto „přehnanou“ úspěšnost můžou způsobovat trénovací data, která jsem použil jak pro testování, tak i pro nastavení aplikace. Je zde dobře vidět, že tato metoda založená na rozpoznání vzorů jde nastavit tak, aby výsledek klasifikace byl velice dobrý (nezáleží na doméně).

	Dovolená	**Diskuse**	**Technické údaje**
P. přesnost	0,955	0,916	0,875
N. přesnost	0,666	1	1
Úplnost	0,84	1	1
F-míra	0,894	0,956	0,933

Tabulka 20: Klasifikace souhrnné výsledky

9 Závěr

Diplomová práce se zabývá problematikou automatické detekce vybraných vzorů na webové stránce a následného výpočtu podobnosti s vektorem dotazu. Vektor dotazu popisuje vybranou doménu. Aplikace dokáže zařadit webové stránky do kategorie.

V úvodu práce jsem se zaměřil na teoretický úvod do rozsáhlé problematiky. V kapitole 3 jsem uvedl modely, které jsou použitelné pro popis dokumentů. Jedná se o model booleovský a vektorový. Moje práce používá výhradně model vektorový. V kapitole 4.1 jsem popisoval obecnou techniku dolování dat z databáze, na kterou navazuje kapitola 4.2 text mining, která se zabývá přípravou a zpracováním textu. Poslední teoretickou částí je kapitola 4.3 web mining. Web mining využívá techniky data miningu a někdy bývá označen jako text mining, protože se zabývá hlavně textem, který „vydoluje“ na webových stránkách. V této kapitole je popsán princip, jak využívat webové vzory ke klasifikaci (tzv. Gestalt principy).

Po teoretickém základu následuje kapitola 5.1 metodika sběru dat. V této kapitole jsem popsal: kde jsem získal data, v jakém formátu jsem je uložil a jaké jsem vybral kódování. Takto nashromážděná data jsem použil pro testování i pro nastavení aplikace. V další kapitole 5.2 jsem uvedl metodiku, jak jsem analyzoval data. Popisuji zde postup, který byl identický pro všechny analyzované domény. Hlavním cílem je nalézt bloky textu, které se na stránkách stále opakují(webový vzor). Tyto bloky je nutné analyzovat - vyjádření pomocí klíčových slov, datových typů a zaznamenání v jakém TAGu stránky se text nachází. Následuje kapitola 5.3, ve které je popsáno uložení konfiguračních souborů a popis jednotlivých atributů. Konfigurace je přehledně uložena do XML souborů.

Po kapitolách, které se týkají vlastní metody řešení, následuje část implementace a testování. V kapitole 6 popisuji všechny použité metody a algoritmy. U některých uvádím zdrojový kód, který je buď přímo v jazyku C#, nebo je napsán v pseudokódu. Nejpodstatnější části algoritmu jsou: rozdělení textu na tokeny, určení datových typů a analýza připraveného textu. V poslední části práce vyhodnocuji výsledky. Prováděl jsem dva druhy testů.

První z nich se týká porovnání výsledků automatické a ruční detekce. Předmětem porovnání je jeden vzor. Porovnání provádím pomocí matice záměn. Matice si pamatuje správné kladné predikce (respektive správné záporné) a falešné kladné predikce (respektive falešné záporné). Z těchto hodnot se dá vypočítat mnoho parametrů, kde hlavním z nich je souhrnná f-míra (kombinuje přesnost a úplnost). Jako druhý testuji klasifikátor, který o každé stránce dokáže říci, jak moc patří do testované domény.

V doméně *Dovolená* jsem vzory *Titulek* a *Teplota* nalezl bez chyby (f-míra = 1). Vzor *Nabídka dovolené* aplikace nalezla s menší chybou (f-míra = 0,895), tato chyba není závažná

a dá se říci že vzor byl velmi dobře nalezen. U vzoru *Vyhledání dovolené* je f-míra = 0,878 na základě této hodnoty mohu říci, že vzor byl také velmi dobře nalezen. Klasifikátor pro doménu *Dovolená* byl otestován a výsledkem byla f-míra = 0,894. Z hodnoty je patrné, že klasifikátor správně zařadil většinu testovaných stránek.

V doméně *Diskuse* jsem vzor *Titulek* nalezl zcela správně (f-mira = 1). Vzor *Příspěvek* byl nalezen s chybou f-míra = 0,811. Pokud f-míra neklesne pod 0,8, není chybovost tak velká, a dá se říci, že vzor byl vyhledán dobře. Klasifikátor pro doménu *Diskuse* funguje bez jediné chyby. Klasifikátor dokázal správně zařadit všechny testované stránky.

V doméně *Technické údaje* jsem vzor *Titulek* opět nalezl bez jediné chyby. Tato opakovaná správnost je způsobena tím, že titulek je velice jednoduchý (obvykle je to jedna věta umístěná v jediném TAGu a neobsahuje žádné datové typy.). Není těžké analyzovat, zda obsahuje či neobsahuje určité slovo. Vzor *Technický údaj* aplikace nalezla s největší chybou (f-míra = 0,737) tato chyba není tak velká, aby testovaná metoda byla neúspěšná. Tato největší chyba je způsobena rozmanitostí technických údajů. Klasifikace proběhla velice dobře, konkrétně s hodnotou f-míra = 0,933.

Z těchto výsledků je patrné, že metoda založená na vyhledávání návrhových vzorů je funkční a dá se použít pro klasifikaci webových stránek. Tato metoda se dá použít pro zpřesnění výsledků běžných internetových vyhledávačů (analýzou relevantních stránek, které internetový vyhledávač již předložil). Největším problémem je nutnost ručně analyzovat co nejvíce dat a stanovit návrhové vzory, které kvalitně popisují doménu.

10 Reference

[1] SMIČKA, Radim. *Optimalizace pro vyhledávače - SEO: Jak zvýšit návštěvnost webu*. 1. vyd. Dubany: Jaroslava Smičková, 2004. 120 s. ISBN: 80-239-2961-5.

[2] Pospíšil Jaromír, Nemrava Michal. *Dolování dat a jeho aplikace*. [online]. 2006. Ver. 1. [cit. 2011-1-4]. Dostupné z URL: <http://axpsu.fpf.slu.cz/~sos10um/trendy/DM.pdf>.

[3] Procházka Michal. *Data mining: jiný pohled na problém*. [online]. Poslední úpravy 29. 11. 2010. [cit. 2011-1-4]. Dostupné z URL: <http://vtm.zive.cz/aktuality/data-mining-jiny-pohled-na-problem>.

[4] Jiawei Han, Micheline Kamber *Data Mining Concepts and Techniques* 2. vyd. San Francisco: Morgan Kaufmann, 2006. 743 s. ISBN 13:978-1-55860-901-3, ISBN 10:1-55860-901-6.

[5] Fayyad Usama M. *Data Mining and Knowledge Discovery.* An International Journal. [online]. 1996. Ver. 1. [cit. 2010-1-4]. Dostupné z URL: <http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf>.

[6] *CRoss Industry Standard Process for Data Mining*. [online]. [cit. 2011-1-4]. Dostupné z URL:<http://www.crisp-dm.org/Process/index.htm>.

[7] Sedláček Petr. *Text mining a jeho možnosti (aplikace)*. Fakultra informatiky MU. [online]. 2003. [cit. 2011-1-4]. Dostupné z URL: <http://www.fi.muni.cz/usr/jkucera/pv109/2003p/xsedlac5.htm>.

[8] Vítek Martin. *Dolování z textu* Brno, FIT VUT Brno. [online]. 2009. [cit. 2011-2-4]. Dostupné z URL: http://www.fit.vutbr.cz/study/courses/ZZD/public/seminar0405/vitek.pdf.

[9] Hroza Jiří. *Automatická podpora filtrace elektronických textových dokumentů metodami strojového učení* Brno, Fakulta informatiky MU. [online]. 2004. [cit. 2011-2-4]. Dostupné z URL: http://is.muni.cz/th/3800/fi_r/rigo_hroza.pdf.

[10] Večeřová Petra. *Předzpracování textových dat pro relační dolování*. Brno, 2009. 47 s. Diplomová práce na fakultě Informatiky Uneverzity Masarykovy. Vedoucí práce Lubomír Popelínský.

[11] Leixner Petr. *Shlukování textových dat*. Brno, 2010. 71 s. Diplomová práce na fakultě Informačních technologií na Univerzitě VUT Brno. Vedoucí práce Vladimír Bartík.

[12] Sedláček Radek. *Morfologický analyzátor češtiny*. Brno, 1999. 78 s. Diplomová práce na fakultě Informatiky Univerzity Masarykovy. Vedoucí práce Pavel Rychlý.

[13] Markov Zdravko, Daniel T. Larose. *Data-mining the Web: uncovering patterns in Web content, structure, and usage.* New Jersey, John Wiley & Sons, Inc., 2007.

[14] Krátký Michal. *Využití SVD pro indexování latentní sémantiky*. Computer Science, VŠB-Technical University of Ostrava. 20 s.

[15] Major Martin. *Klasifikace obsahu dokumentů*. Praha, 2009. 37 s. Bakalářská práce na Matematicko-fyzikální fakulte Univerzity Karlovy v Praze. Vedoucí Martin Mareš.

[16] Dušan Húsek, Hana Řezanková, Václav Snášel. *Shlukování a textové dokumenty* 2004, 8 s. JČMF 2004.

[17] Petr Berka. *Dobývání znalostí z databází*. Praha : Academia. Kapitola Shluková analýza, 366 s. ISBN 80-200-1062-9.

[18] Pavel Berkhin. *Survey of Clustering Data Mining Techniques*, 2002.

[19] Pokorný J., Snášel V., Kopecký M. *Dokumentografické informační systémy*. Karolinum, Skriptum MFF UK Praha, 2005, ISBN 80-246-1148-1.

[20] Húsek D.,Pokorný J., Snášel V., Řezanková H. *Metody vyhledávání v rozsáhlých kolekcí dat*, Brno, 2003, Datakon. 30 s.

[21] Svobodová Kristýna *Data Mining*, [online]. [cit. 2011-7-4]. Poslední úpravy 13.6.2010. Dostupné z URL: `http://kisk.phil.muni.cz/wiki/Data_Mining`.

[22] Ricardo Baeza-Yates, Berthier Ribeiro-Neto. *Modern Information Retrieval*, 2 vyd., 1999, ACM press New York, ISBN-13: 978-0-201-39829-8.

[23] Miloš Kudělka, Václav Snášel, Obdřej Lehečka, Eyas El-Qawasmeh. *Web Content Mining Using Web Design Patterns*, 2008, 6 s.

[24] Miguel Gomes da Costa, Zhiguo Gong. *Structure Mining: An Introduction*, 2005, 6 s, International Conference on Information Acquisition.

[25] Lukáš Čenovský. *Web Usage Mining on is.muni.cz* Diplomová práce na fakultě informatiky University Masarykovy v Brně.

[26] Jiří Jelínek. *Uživatelská podpora v prostředí WWW*, Inforum 2004, Praha 26.5.2004. 14 s.

[27] Raymond Kosala, Hendrik Blockeel. *WebMining Research: A Survey*, ACM SIGKDD, July 2000. 15s.

A Zdrojové kódy

A.1 Rozdělení na tokeny

```
for (int i = 0; i < delkaVety; i++)
{
    // jestlize je velke
    if (char.IsUpper(s[i]) )
    {
        p += s[i];
        //pokud je dalsi velke,znamena to ze slovo bude napsane velkym pismenem
        if (i + 1 < de && char.IsUpper(s[i + 1]))
        {
            i++;
            while (i < delkaVety && char.IsUpper(s[i]))
            {
                p += s[i];
                i++;
            }
            i--;
            if (i + 1 < delkaVety && char.IsWhiteSpace(s[i + 1]))
            { slova += p + s[i + 1]; p = null; i++; continue; }
            else
            {
                slova += p + "␣";
            }
            p = null; continue;
        }
        i++;
        //pokud je dalsi male,znamena to ze slovo je napsane s pocatecnim velkym pismenem
        while (i < delkaVety && char.IsLower(s[i]))
        {
            p += s[i];
            i++;
        }
        i--;
        //pokud je dalsi znak mezera vlozim na konec a pokracuji
        if (i + 1 < delkaVety && char.IsWhiteSpace(s[i + 1]))
        { slova += p + s[i + 1]; p = null; i++; continue; }
        //pokud je to neaky znak vlozim oddelovaci mezeru a pokracuji
        else
        {
            slova += p + "␣";
        }
        p = null; continue;
```

```
    }
    if (char.IsLower(s[i]) )
    {
        //kod pro malé
    }
    if (char.IsDigit(s[i]) )
    {
        //kod por cislovky
    }
    // jestlize je to mezera a neni to pocatek vstupniho retezce a predchozi znak nebyl mezera, tak
        vlozim mezeru
    if (char.IsWhiteSpace(s[i]) && slova != null && !char.IsWhiteSpace(slova[slova.Length - 1]))
        slova += s[i];
    //pokud je to znak a neni to mezera, vlozim znak a za nej oddelovaci mezeru
    if (!char.IsLetterOrDigit(s[i]) && !char.IsWhiteSpace(s[i])) slova += s[i] + "␣";
}
```

Výpis 15: Tokenizace

A.2 Analýza 2

```
1  List<string> Analyza(zdroj, text, maxVzdálenost, minPočetNalezených, musiObsahovat)
2  FOREACH (pro každé klíčové slovo s˜se vzoru)
3      VLOZ s˜DO klíčová slova
4  FOR (kazde slovo se zdroj proveď)
5      IF (slovo == typ text) THEN
6          odstraň koncovku z slovo
7          FOREACH (pro každé KS z˜klíčové slova)
8              odsran koncovku z KS
9              IF (slovo == KS) THEN
10                 IF (je to první výskyt slova) THEN
11                     nastav první a poslední pozici
12                     VLOZ slovo do pomocné
13                 ELSE THEN
14                     IF (pokud je vzdálenost mezi prvním a posledním slovem <= maxVzdálenost) THEN
15                         IF (pomocné neobsahuje slovo) THEN
16                             VLOZ slovo do pomocné
17                             NASTAV pozici
18                     IF (počet slov v˜pomocné >= minPocetNalezenych && pomocné obsahuje povinné)
                           THEN
19                         vytvoř vetu se zdroj[první ... poslední pozice]
20                         do výsledek vlož pomocné a vetu
21                         nastav první a poslední pozici
22                         vymaz pomocné
```

```
23                  IF (pomocné neobsahuje povinné && vzdálenost <= maxVzdálenost && počet v
                        pomocné >= minPocetNalezenych) THEN
24                    IF (pomocne neobsahuje slovo) THEN
25                    VLOZ slovo do pomocne
26                    nastav pozici
27                  ELSE THEN
28                    ODEBER první pozici z pomocné
29                    nastav pozici
30          ELSE THEN
31            obdobně pro datové typy
32      IF (vysledek.Count > 0) THEN
33        do vzoru přičti počet záznamů ve výsledek
34      RETURN výsledek
```

Výpis 16: Analýza 2

B Popis vzoru (XML)

B.1 Vzor dovolená

```
<?xml version="1.0" encoding="Windows-1250"?>
<vzor>
  <jmeno>Dovolená</jmeno>
  <segment id = "titulek">
    <seg_hodnota>1</seg_hodnota>
    <seg_maxVelikost>999</seg_maxVelikost>
    <seg_minVelikost>10</seg_minVelikost>
    <seg_minPocetNalezenych>1</seg_minPocetNalezenych>
    <seg_maxVzdalenostSlov>-1</seg_maxVzdalenostSlov>
    <seg_musiObsahovat>null</seg_musiObsahovat>
    <seg_slovo>dovolena</seg_slovo> <seg_slovo>zajezd</seg_slovo>
    <seg_tag>title</seg_tag>
  </segment>
  <segment id = "teplota">
    <seg_hodnota>2,5</seg_hodnota>
    <seg_maxVelikost>1500</seg_maxVelikost>
    <seg_minVelikost>5</seg_minVelikost>
    <seg_minPocetNalezenych>1</seg_minPocetNalezenych>
    <seg_maxVzdalenostSlov>4</seg_maxVzdalenostSlov>
    <seg_musiObsahovat>null</seg_musiObsahovat>
    <seg_token>Tteplota</seg_token>
    <seg_tag>tr</seg_tag> <seg_tag>ul</seg_tag>
    <seg_tag>span</seg_tag> <seg_tag>div</seg_tag>
  </segment>
  <segment id = "nabidka_dovolene">
    <seg_hodnota>3,5</seg_hodnota>
    <seg_maxVelikost>3000</seg_maxVelikost>
    <seg_minVelikost>15</seg_minVelikost>
    <seg_minPocetNalezenych>3</seg_minPocetNalezenych>
    <seg_maxVzdalenostSlov>18</seg_maxVzdalenostSlov>
    <seg_musiObsahovat>Tdatum</seg_musiObsahovat>
    <seg_musiObsahovat>Tcena</seg_musiObsahovat>
    <seg_slovo>dnu</seg_slovo> <seg_slovo>snidane</seg_slovo>
    <seg_slovo>bez</seg_slovo> <seg_slovo>polopenze</seg_slovo>
    <seg_slovo>dovolena</seg_slovo> <seg_slovo>ostrov</seg_slovo>
    <seg_slovo>obednat</seg_slovo> <seg_slovo>bus</seg_slovo>
    <seg_slovo>letecky</seg_slovo> <seg_slovo>vlastni</seg_slovo>
    <seg_token>Tdatum</seg_token> <seg_token>Tcena</seg_token>
    <seg_token>Tdestinace</seg_token> <seg_token>TmaleCislo</seg_token>
    <seg_token>Tteplota</seg_token> <seg_tag>tr</seg_tag>
    <seg_tag>a</seg_tag> <seg_tag>ul</seg_tag>
```

```
    <seg_tag>span</seg_tag> <seg_tag>div</seg_tag>
  </segment>
  <segment id = "vyhledani_dovolene">
    <seg_hodnota>1</seg_hodnota>
    <seg_maxVelikost>3500</seg_maxVelikost>
    <seg_minVelikost>250</seg_minVelikost>
    <seg_minPocetNalezenych>4</seg_minPocetNalezenych>
    <seg_maxVzdalenostSlov>-1</seg_maxVzdalenostSlov>
    <seg_musiObsahovat>null</seg_musiObsahovat>
    <seg_slovo>vyhledavani</seg_slovo> <seg_slovo>vyberte</seg_slovo>
    <seg_slovo>oblast</seg_slovo> <seg_slovo>zeme</seg_slovo>
    <seg_slovo>cena</seg_slovo> <seg_slovo>termin</seg_slovo>
    <seg_slovo>datum</seg_slovo> <seg_slovo>odjezd</seg_slovo>
    <seg_slovo>hledat</seg_slovo> <seg_slovo>odeslat</seg_slovo>
    <seg_slovo>dotaz</seg_slovo> <seg_slovo>delka</seg_slovo>
    <seg_slovo>nerozhoduje</seg_slovo> <seg_slovo>kategorie</seg_slovo>
    <seg_slovo>pocet</seg_slovo> <seg_slovo>doprava</seg_slovo>
    <seg_slovo>rozsirene</seg_slovo> <seg_slovo>prijezd</seg_slovo>
    <seg_slovo>minimalne</seg_slovo> <seg_slovo>lokalita</seg_slovo>
    <seg_slovo>lm</seg_slovo>
    <seg_tag>fieldset</seg_tag> <seg_tag>table</seg_tag>
    <seg_tag>div</seg_tag>
  </segment>
</vzor>
```

Výpis 17: Vzor domény dovolená

B.2 Vzor diskuze

```
<?xml version="1.0" encoding="Windows-1250"?>
<vzor>
  <jmeno>Diskusní fórum</jmeno>
  <segment id = "titulek">
    <seg_hodnota>1</seg_hodnota>
    <seg_maxVelikost>999</seg_maxVelikost>
    <seg_minVelikost>10</seg_minVelikost>
    <seg_minPocetNalezenych>1</seg_minPocetNalezenych>
    <seg_maxVzdalenostSlov>-1</seg_maxVzdalenostSlov>
    <seg_musiObsahovat>null</seg_musiObsahovat>
    <seg_slovo>diskusni</seg_slovo> <seg_slovo>forum</seg_slovo>
    <seg_tag>title</seg_tag>
  </segment>
  <segment id = "prispevek">
    <seg_hodnota>4</seg_hodnota>
```

```
    <seg_maxVelikost>1550</seg_maxVelikost>
    <seg_minVelikost>70</seg_minVelikost>
    <seg_minPocetNalezenych>3</seg_minPocetNalezenych>
    <seg_maxVzdalenostSlov>−1</seg_maxVzdalenostSlov>
    <seg_musiObsahovat>Tdatum</seg_musiObsahovat>
    <seg_slovo>info</seg_slovo> <seg_slovo>re</seg_slovo>
    <seg_slovo>od:</seg_slovo> <seg_slovo>predmet</seg_slovo>
    <seg_slovo>offline</seg_slovo> <seg_slovo>|</seg_slovo>
    <seg_slovo>member</seg_slovo> <seg_slovo>host</seg_slovo>
    <seg_slovo>clen</seg_slovo> <seg_slovo>moderator</seg_slovo>
    <seg_slovo>prispevky</seg_slovo> <seg_slovo>zalozen</seg_slovo>
    <seg_slovo>registrovan</seg_slovo> <seg_slovo>odpovedet</seg_slovo>
    <seg_slovo>reagovat</seg_slovo> <seg_slovo>zaslano</seg_slovo>
    <seg_slovo>pravidel</seg_slovo> <seg_slovo>nahoru</seg_slovo>
    <seg_slovo>napsano</seg_slovo> <seg_slovo>napsal</seg_slovo>
    <seg_slovo>nevhodny</seg_slovo> <seg_slovo>vlozeno</seg_slovo>
    <seg_slovo>ip</seg_slovo> <seg_token>Tdatum</seg_token>
    <seg_token>TporadoveCislo</seg_token>
    <seg_tag>div</seg_tag> <seg_tag>table</seg_tag>
    <seg_tag>tr</seg_tag> <seg_tag>li</seg_tag>
    <seg_tag>p</seg_tag>
  </segment>
</vzor>
```

Výpis 18: Vzor domény diskuze

B.3 Vzor technické údaje

```
<?xml version="1.0" encoding="Windows−1250"?>
<vzor>
  <jmeno>Technické údaje</jmeno>
  <segment id = "titulek">
    <seg_hodnota>5</seg_hodnota>
    <seg_maxVelikost>999</seg_maxVelikost>
    <seg_minVelikost>10</seg_minVelikost>
    <seg_minPocetNalezenych>1</seg_minPocetNalezenych>
    <seg_maxVzdalenostSlov>−1</seg_maxVzdalenostSlov>
    <seg_musiObsahovat>null</seg_musiObsahovat>
    <seg_slovo>popis</seg_slovo> <seg_slovo>technicke</seg_slovo>
    <seg_slovo>specifikace</seg_slovo> <seg_slovo>udaje</seg_slovo>
    <seg_slovo>parametry</seg_slovo>
    <seg_tag>title</seg_tag>
  </segment>
  <segment id = "technicky_popis">
```

<seg_hodnota>10</seg_hodnota>
<seg_maxVelikost>10000</seg_maxVelikost>
<seg_minVelikost>40</seg_minVelikost>
<seg_minPocetNalezenych>3</seg_minPocetNalezenych>
<seg_maxVzdalenostSlov>6</seg_maxVzdalenostSlov>
<seg_musiObsahovat>Tjednotka</seg_musiObsahovat>
<seg_musiObsahovat>cislo</seg_musiObsahovat>
<seg_slovo>baterie</seg_slovo> <seg_slovo>displej</seg_slovo>
<seg_slovo>disk</seg_slovo> <seg_slovo>digitalni</seg_slovo>
<seg_slovo>doba</seg_slovo> <seg_slovo>dosah</seg_slovo>
<seg_slovo>rozmery</seg_slovo> <seg_slovo>hmotnost</seg_slovo>
<seg_slovo>objem</seg_slovo> <seg_slovo>odezva</seg_slovo>
<seg_slovo>obraz</seg_slovo> <seg_slovo>opticky</seg_slovo>
<seg_slovo>otacek</seg_slovo> <seg_slovo>ohnisko</seg_slovo>
<seg_slovo>obvod</seg_slovo> <seg_slovo>teplota</seg_slovo>
<seg_slovo>rozliseni</seg_slovo> <seg_slovo>rozlisenim</seg_slovo>
<seg_slovo>rychlost</seg_slovo> <seg_slovo>rozvor</seg_slovo>
<seg_slovo>rozchod</seg_slovo> <seg_slovo>rozpeti</seg_slovo>
<seg_slovo>ram</seg_slovo> <seg_slovo>rom</seg_slovo>
<seg_slovo>pamet</seg_slovo> <seg_slovo>makrem</seg_slovo>
<seg_slovo>mezipamet</seg_slovo> <seg_slovo>mezimesto</seg_slovo>
<seg_slovo>megapixel</seg_slovo> <seg_slovo>nadrz</seg_slovo>
<seg_slovo>napeti</seg_slovo> <seg_slovo>procesor</seg_slovo>
<seg_slovo>prehravani</seg_slovo> <seg_slovo>pasmu</seg_slovo>
<seg_slovo>plocha</seg_slovo> <seg_slovo>prostor</seg_slovo>
<seg_slovo>proud</seg_slovo> <seg_slovo>pripojeni</seg_slovo>
<seg_slovo>pomer</seg_slovo> <seg_slovo>pojezd</seg_slovo>
<seg_slovo>vlastnost</seg_slovo> <seg_slovo>format</seg_slovo>
<seg_slovo>frekvence</seg_slovo> <seg_slovo>fotoaparat</seg_slovo>
<seg_slovo>spotreba</seg_slovo> <seg_slovo>standardne</seg_slovo>
<seg_slovo>stereo</seg_slovo> <seg_slovo>stupnu</seg_slovo>
<seg_slovo>strana</seg_slovo> <seg_slovo>sila</seg_slovo>
<seg_slovo>ventil</seg_slovo> <seg_slovo>vykon</seg_slovo>
<seg_slovo>vaha</seg_slovo> <seg_slovo>vazi</seg_slovo>
<seg_slovo>vyska</seg_slovo> <seg_slovo>vysuv</seg_slovo>
<seg_slovo>tocivy</seg_slovo> <seg_slovo>tlak</seg_slovo>
<seg_slovo>max</seg_slovo> <seg_slovo>min</seg_slovo>
<seg_slovo>kapacita</seg_slovo> <seg_slovo>konektor</seg_slovo>
<seg_slovo>kroutici</seg_slovo> <seg_slovo>delka</seg_slovo>
<seg_slovo>sirka</seg_slovo> <seg_slovo>hloubka</seg_slovo>
<seg_slovo>uhlopricka</seg_slovo> <seg_slovo>uhel</seg_slovo>
<seg_slovo>uhly</seg_slovo> <seg_slovo>ati</seg_slovo>
<seg_slovo>analogovy</seg_slovo> <seg_slovo>sklon</seg_slovo>
<seg_slovo>nvidia</seg_slovo> <seg_slovo>lan</seg_slovo>
<seg_slovo>ethernet</seg_slovo> <seg_slovo>eletricke</seg_slovo>

```
    <seg_token>TmaleCislo</seg_token> <seg_token>Tcislo</seg_token>
    <seg_token>Tjednotka</seg_token> <seg_slovo>iso</seg_slovo>
    <seg_slovo>sata</seg_slovo> <seg_slovo>ide</seg_slovo>
    <seg_slovo>dohc</seg_slovo> <seg_slovo>sub</seg_slovo>
    <seg_slovo>sd</seg_slovo> <seg_slovo>dvd</seg_slovo>
    <seg_slovo>cd</seg_slovo> <seg_slovo>zoom</seg_slovo>
    <seg_slovo>zvuk</seg_slovo> <seg_slovo>gtg</seg_slovo>
    <seg_slovo>gprs</seg_slovo> <seg_slovo>edge</seg_slovo>
    <seg_slovo>wcdma</seg_slovo> <seg_slovo>gsm</seg_slovo>
    <seg_slovo>qvga</seg_slovo> <seg_slovo>tft</seg_slovo>
    <seg_slovo>vga</seg_slovo>
    <seg_tag>div</seg_tag> <seg_tag>tr</seg_tag>
    <seg_tag>table</seg_tag> <seg_tag>ul</seg_tag>
    <seg_tag>p</seg_tag>
  </segment>
</vzor>
```

Výpis 19: Vzor domény technické údaje

C Příručky

Programátorskou příručku jsem vygeneroval ze zdrojového kódu. Pro vygenerování jsem použil program Microsoft Sandcastle. Příručka je vygenerovaná v přehledném stylu MSDN. Programátorská příručka je umístěna na CD ve složce PRILOHY/Programátorská příručka.
Uživatelská příručka je umístěna na CD ve složce PRILOHY/Uživatelská příručka